Proyecto Fin de Carrera
Ingeniería de Industrial

Herramienta para el cálculo de la superficie requerida en una planta fotovoltaica

Autor: Ana Chacón García
Tutor: Isidoro Lillo Bravo

Dpto. Ingeniería Energética
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Carrera
Ingeniería Industrial

Herramienta para el cálculo de la superficie requerida en una planta fotovoltaica

Autora:
Ana Chacón García

Tutor:
Isidoro Lillo Bravo
Profesor Contratado Doctor

Dpto. de Ingeniería Energética
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Proyecto Fin de Carrera: Herramienta para el cálculo de la superficie requerida en una planta fotovoltaica

Autor: Ana Chacón García

Tutor: Isidoro Lillo Bravo

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2018

El Secretario del Tribunal

Índice

Notación

α	Altura solar
β	Acimut solar
δ	Declinación solar
φ	Latitud
λ	Longitud
ω	Ángulo horario
ω_s	Ángulo horario de salida y puesta del sol.
D	Distancia entre filas de módulos
D1	Distancia ocupada por una fila de módulos
D2	Sombra máxima producida por una fila de módulos
H	Altura de una fila de módulos
h1	Altura relativa entre paneles
h2	Diferencia del nivel del terreno entre subconjuntos
K	Ancho ocupado por una fila de módulos, o longitud total de las filas de un subconjunto.
L	Largo de una fila de módulos, o longitud total de las columnas de un subconjunto.
L'	Proyección horizontal de los rayos solares entre dos subconjuntos de módulos.
N	Número de días contados desde el 1 de enero
Z	Ángulo de inclinación de un panel fotovoltaico
γ	Acimut del conjunto de paneles
α	Altura solar
β	Acimut solar
δ	Declinación solar
φ	Latitud
λ	Longitud
ω	Ángulo horario
ω_s	Ángulo horario de salida y puesta del sol.
D	Distancia entre filas de módulos
D1	Distancia ocupada por una fila de módulos
D2	Sombra máxima producida por una fila de módulos

1 INTRODUCCIÓN

Actualmente se disponen de numerosos programas para el diseño de plantas fotovoltaicas, que permiten al usuario optimizar la producción de energía de una instalación. A menudo las instalaciones pequeñas se diseñan de forma manual, pero uno de los parámetros a tener en cuenta en el diseño una instalación, grande o pequeña, es la distancia que es necesario separar los subconjuntos de módulos entre sí para que no se produzcan sombreados entre los paneles. Si se producen sombreados se generarán importantes pérdidas en la producción de energía, y si se separan en exceso se desaprovechará parte del terreno destinado a la instalación. En este trabajo se creará una herramienta que facilite el cálculo de la distancia necesaria que se deben separar los módulos para que no se produzcan sombreados entre subconjuntos o mesas de módulos.

Se facilitará el código fuente de una herramienta para calcular tanto la distancia entre paneles como del área total ocupada por subconjunto. La herramienta puede ser usada de tres maneras diferentes: a través de una interfaz gráfica, en que el usuario puede introducir los datos de entrada necesarios, o mediante la línea de comandos, en cuyo caso el usuario podrá proporcionar los datos directamente o a través de un archivo.

Posteriormente se usará esta herramienta para plantear diversas situaciones, considerando varias configuraciones de plantas fotovoltaicas. Se calcularán las distancias necesarias entre subconjuntos de las situaciones y se planteará la realización de este cálculo para obtener el área total requerida para la instalación que se desee.

2 OBJETIVOS

El propósito de este trabajo es participar en la mejora de la situación actual hacia la optimización de producción de la energía eléctrica. Se busca crear una herramienta en lenguaje de programación Python que automatiza el cálculo de la distancia necesaria entre subconjuntos de módulos para que no se produzcan sombreados de paneles en una instalación fotovoltaica bajo las condiciones de contorno deseadas. La intención es que herramienta que sea fácil de usar, apta para todo tipo de usuarios, desde aquellos que desean hacer un cálculo manual de la instalación, como para la posible adición de esta herramienta a programas ya generados para el diseño de instalaciones fotovoltaicas.

3 ESTADO DEL ARTE

El problema principal de sombras en módulos fotovoltaicos es que la producción de energía se reduce significativamente debido a sombras parciales. Cuando el sol se encuentra bajo en las horas iniciales del día, y al atardecer, y especialmente en aquellas latitudes no cercanas al ecuador, se producen sombras en la parte inferior de los paneles, y el efecto de sombras en una de las células afecta la producción de todo el módulo. Los módulos, formados por conjuntos de células, tendrán tantas pérdidas en su producción como tenga la célula con mayor superficie sombreada [1].

Por ello es necesario definir adecuadamente la distancia a las que se instalarán las diferentes mesas, subconjuntos de módulos (o arrays, en inglés).

Generalmente se instalan los paneles fotovoltaicos a un ángulo superior a cero en terrenos planos y en techos no inclinados, por generar mayor energía y menos acumulación de suciedad en comparación con instalaciones totalmente horizontales. El inconveniente al instalar los paneles en un ángulo superior a cero son las sombras producidas por los paneles entre sí, ya que los sistemas fotovoltaicos son altamente sensibles a sombras parciales y disminuye la producción de energía y por tanto el rendimiento obtenido de la instalación.

Al optimizar la distancia entre subconjuntos, disminuirán las pérdidas por sombra y reducirán las distancias entre subconjuntos con el subsiguiente aprovechamiento del terreno. Esto es especialmente importante en casos de zonas con escasez de terreno disponible para instalaciones fotovoltaicas. De esta forma, se aprovecharán al máximo los recursos con la consiguiente obtención de mayor potencia por metro cuadrado en el terreno usado.

La metodología aplicada para el cálculo varía, y se suele basar en cálculos geométricos sencillos, en los que se tienen en cuenta la latitud y algunos datos de la geometría de la construcción.
Una forma de calcular las distancias entre subconjuntos de módulos en terrenos es aplicar la siguiente relación, en la que la distancia entre paneles es:

$$D = 2.5 \cdot H$$

Con D la distancia entre mesas y H la proyección vertical o altura de los subconjuntos una vez instalados. Esta fórmula se suele usar para latitudes en torno a los 40 grados.

En ocasiones se usa la fórmula:

$$d = H/(\tan(61º - \varphi)$$

con φ la latitud del terreno.

Estas relaciones, aunque prácticas y fácil de aplicar, no tienen en cuenta varios de los parámetros que influyen en las sombras producidas y por tanto se trata de una estimación con posibilidades de mejora.

Por otro lado, existen artículos publicados en los que se investigan maneras más efectivas, con cálculo más detallado y minucioso, pero aún esta información no necesariamente se lleva a la

práctica. Por ejemplo, la referencia [2] se provee el cálculo de las pérdidas por sombra, y se optimiza la distancia entre subconjuntos, pero los cálculos en este caso sólo son aplicables a localizaciones centroeuropeas.

O en el artículo [3] se proporciona un estudio del cálculo de la distancia para suelos inclinados.
En el informe técnico [4], se trabajan con detalle numerosas configuraciones posibles, incluyendo multitud de cálculos para terrenos planos e inclinados, además de cálculo con referencia concreta a la geometría de diferentes soportes de módulos, como pueden ser: soportes fijos, regulables, con seguimiento en un eje de rotación, dos, etc.
El inconveniente del uso de artículos como estos no es más que el cálculo de la distancia de acuerdo con esta información requiere de investigación y comprensión adecuada tanto de geometría solar (que puede llegar a ser relativamente compleja) como matemática, y la realización de un cálculo metódico para poder llegar a un resultado aplicable.

4 MODELO

En esta sección se desarrollará el modelo matemático sobre el que se construye la herramienta, aportando los datos y las fórmulas que se aplicarán. Se desarrolla en la sección primera del capítulo conceptos y definiciones para la adecuada comprensión del modelo, y posteriormente se procede a la aportación de la explicación matemática.

Se tomará como principal referencia el informe técnico 'Mathematic Models and Calculation Examples for Land Usage of PV Farms' [4] que se encuentra actualmente pendiente de publicación pero aporta datos relevantes acerca del cálculo de distancias entre módulos y uso del terreno en granjas fotovoltaicas.

Se tomará en particular las situaciones de módulos fotovoltaicos fijos en terreno plano con orientación sur, en terreno plano con cualquier orientación, y en terreno inclinado con cualquier orientación para este proyecto.

4.1 Términos y definiciones utilizadas.

Se explican a continuación algunos conceptos para la correcta comprensión del modelo matemático.

Latitud (φ): es la distancia angular entre la línea ecuatorial y un punto determinado de la Tierra, medida a lo largo del meridiano en el que se encuentra dicho punto, variando desde 0º en el Ecuador hasta 90º en los polos. Según el hemisferio en el que se sitúa el punto, puede ser latitud norte o sur. Líneas de latitud constante, o paralelos, trazan circunferencias paralelas al ecuador con dirección este-oeste.

Longitud (λ): expresa la distancia angular entre un punto dado de la superficie terrestre y el meridiano que se toma como 0º (meridiano de Greenwich), medida a lo largo del paralelo en el que se encuentra dicho punto, una circunferencia cuyo centro es la intersección del eje de la Tierra con el plano del citado paralelo. Meridianos (líneas trazadas de Polo Norte a Sur), conectan puntos de igual longitud. L Longitud de cualquier otro meridiano distinto al de Greenwich tomará valores hasta +180º hacia el Este y hasta -180º hacia el Oeste.

Declinación (δ): ángulo que forma la línea que une el centro de la Tierra y el centro del Sol con el plano del ecuador celeste, que depende únicamente del tiempo.
El eje terrestre está inclinado 23º26', aunque este valor cambia lentamente a lo largo de miles de años, pero puede considerarse constante para este estudio. En los solsticios, el ángulo de declinación solar alcanza sus máximos, iguales a ±23º26'. Así, el valor de δ en el hemisferio norte será +23º26' en el solsticio de verano en, 0º es los equinoccios y –23º26' en el solsticio de invierno.

Se calcula para cada momento con la ecuación de Cooper, que ofrece un error máximo de 1.5º y la fórmula es

$$\delta = 23{,}45\,\mathrm{sen}(360 * \frac{284+N}{365}),$$

siendo N el número de días contados desde el 1 de enero.

Tiempo solar verdadero (TSV): tiempo basado en el movimiento aparente del Sol en la bóveda celeste. A las 0:00 hora solar verdadera, el Sol atraviesa el meridiano del observador y alcanza la máxima altura sobre el horizonte. El Tiempo Solar Verdadero comienza a contarse a partir del mediodía solar (mitad del día).

Ángulo horario (ω): es una de las coordenadas usadas en el sistema ecuatorial de coordenadas (ECS) para expresar la dirección de un punto en la esfera celestial. Representa el desplazamiento angular del Sol en el plano del ecuador celeste. La equivalencia de este ángulo con el Tiempo Solar Verdadero se expresa de la siguiente manera:

$$\omega = 15 \cdot TSV$$

De esta forma, cada hora transcurrida se corresponde con un ángulo de 15º, resultado de dividir los 360º de recorrido entre 24 horas. Así, por la mañana este ángulo será negativo, y por la tarde positivo.

Ángulo horario de la salida y puesta del sol (ω_s): ángulo horario correspondiente a la aparición y desaparición del Sol en el plano del horizonte. Se calcula con la siguiente fórmula:

$$\omega_s = \mathrm{acos}[(-\tan(\delta)) \cdot \tan(\varphi)]$$

Altura solar (α): ángulo de elevación del sol, es decir, el ángulo entre la dirección del sol y el horizonte. Se calcula con la fórmula:

$$sen(\alpha) = sen(\varphi) \cdot sen(\delta) + \cos(\varphi) \cdot \cos(\delta) \cdot \cos(\omega)$$

Acimut solar (β): es el ángulo medido sobre el horizonte celeste que forman el punto cardinal Norte y la proyección vertical del sol sobre el horizonte del. Este parámetro junto con la altura solar permite definir la posición del Sol. El acimut hacia el Este es negativo y hacia el Oeste positivo. Se calcula con la fórmula:

$$sen\ (\beta) = \cos(\delta) \cdot sen\ (\omega)\ /\ \cos(\alpha)$$

ó

$$\cos(\beta) = \left(sen\ (\varphi) \cdot sen\ (\alpha) - sen\ (\delta)\right) / (\cos(\alpha)\ \cdot \cos(\varphi)$$

para β entre 0º y 180º

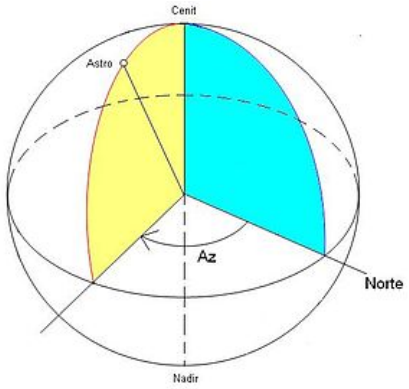

Figura 4.1. Representación del acimut, Az. Fuente: Wikipedia

Acimut de un panel solar, módulo o subconjunto de módulos (γ): ángulo entre una línea trazada con orientación sur (en una localización ubicada en el hemisferio norte, o norte si está ubicada en el hemisferio sur) y la sombra producida en el suelo por la normal del módulo o subconjunto. Si el ángulo es cero, el módulo o subconjunto estará orientado al sur en el hemisferio norte (o al norte en el hemisferio sur). El ángulo será positivo si la orientación es sureste y negativo si es suroeste, o viceversa si se trata del hemisferio sur.

Ángulo de inclinación del módulo (Z): Ángulo entre la horizontal y el plano del módulo.

Distancia sin sombra (D): En la situación en la que dos subconjuntos de módulos están situados uno detrás de otro, en paralelo, se define como la distancia mínima de separación para la que el subconjunto que se encuentra detrás no experimente sombra producida por el subconjunto situado delante suya.

Área que ocupa un subconjunto de paneles: Generalmente se define el área ocupada como la que necesita un conjunto de paneles para su instalación, pero en este contexto generalmente se referirá a la suma de esta más área máxima de sombra generada por el subconjunto. Si se coloca algún elemento dentro de este área se producirán sombras en el mismo.

Porcentaje del día: porcentaje aplicado sobre el total de horas de luz del día (desde que sale el sol hasta que se pone).

Porcentaje del día sin sombra: Porcentaje del número de horas de luz de un día, durante las cuales cual no habrá sombras producidas por un subconjunto de módulos sobre un grupo de módulos situado inmediatamente tras él.

4.2 Condiciones de contorno del modelo.

Se presentan las siguientes condiciones de contorno:

Se realizarán los cálculos para el día con las condiciones de sombra más desfavorable del año, es decir, aquel en el que la duración del día es la mínima del año, la altitud del sol es la mínima en el cielo y por tanto las sombras producidas son más largas. Esto se dará en el solsticio de invierno, instante en que la posición del sol se encuentra a la mayor distancia del ecuador, lo cual ocurre entre el 20 y el 23 de diciembre para el hemisferio norte, y entre el 20 y el 23 de junio para el hemisferio sur.

Para el modelo, se constituyen subconjuntos de módulos (o arrays en inglés) a partir de módulos fotovoltaicos, y se colocarán en el terreno formando filas paralelas entre sí. Cada subconjunto de módulos se encontrará en suelo plano, pero se permite plantear la situación en la que el suelo esté inclinado y haya diferencia de nivel entre el suelo de subconjuntos, para lo cual se allanaría el terreno para colocar los subconjuntos, resultando en escalones o terrazas.

Los subconjuntos tendrán las mismas dimensiones y estarán igualmente orientados e inclinados. Se define la geometría entre dos de los subconjuntos, y si se mantienen las características, los resultados se pueden extender al resto de subconjuntos de la planta.

Se define un porcentaje del día sin sombra, que se establece en el informe técnico como el 75% de la duración del día del solsticio de invierno (porcentaje aplicado sobre el número de horas de sol), aunque para este proyecto se variará este parámetro. Durante este periodo sin sombra se podrá asegurar que no se producen sombreado mutuo en ningún punto los módulos, como se indicó en el apartado anterior. Este porcentaje se puede modificar, para así ajustarse a las condiciones de diseño de cualquier proyecto; a mayor porcentaje del día se defina sin sombreado entre módulos, mayor será la distancia necesaria entre módulos, y se aprovechará en menor medida el terreno, pero se conseguirá una mayor producción de energía.

4.3 Modelo

Se considerarán los módulos o paneles fotovoltaicos, en sus dimensiones de ancho y largo. Al configurar las filas de módulos, se calcularán los parámetros de largo de fila, L y ancho, K. En la siguiente figura se muestra dicha representación con más detalle.

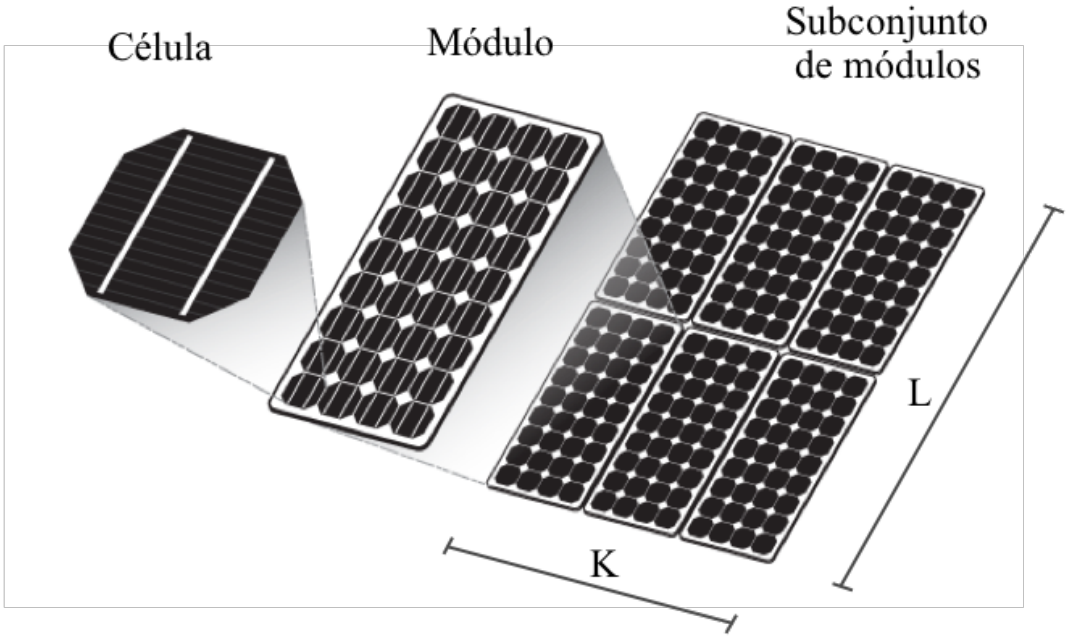

Figura 4.2. Célula, módulo y fila de módulos fotovoltaicos, y medida de ancho (K) y largo del subconjunto de módulos.

El informe se basa en la siguiente figura para realizar los cálculos matemáticos que hacen posible el cálculo de distancias entre filas de subconjuntos.

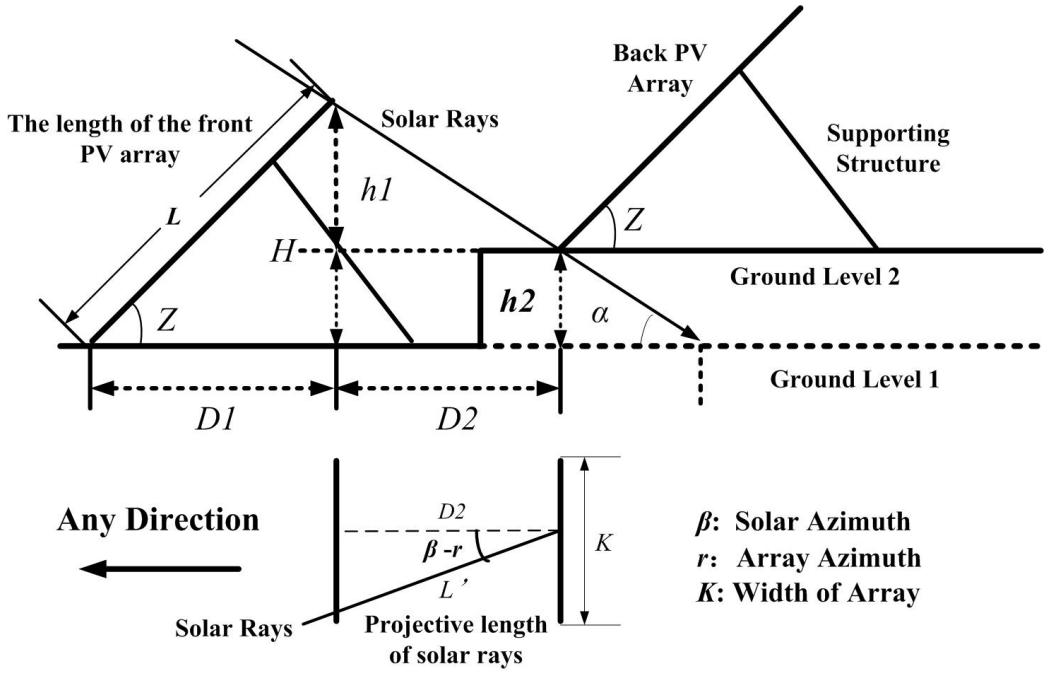

Figura 4.3. Relación entre los rayos solares y filas de módulos fotovoltaicos. Fuente [4]

Donde:

L es el largo del subconjunto
K es el ancho del subconjunto
H es la proyección vertical del subconjunto de módulos
h1 es la altura relativa entre subconjuntos
h2 es la diferencia de altura del terreno
L' es la proyección horizontal de los rayos solares entre dos filas de módulos
Z es el ángulo de inclinación de los paneles
α es la altura solar.
β es el acimut solar.
γ es el acimut del conjunto de paneles
D1 Distancia ocupada por un subconjunto
D2 Sombra máxima producida por un subconjunto

El objeto de los cálculos será la obtención de D1 y D2, de forma que se obtenga la distancia deseada (D) para que no haya sombras en ningún punto del subconjunto posterior durante el periodo de tiempo definido.

De la figura anterior se obtiene:

$$D1 = L \cdot \cos(Z)$$

$$H = L \cdot sen\,(Z)$$

$$H = h1 + h2$$

$$L' = \frac{h1}{\tan \alpha}$$

$$D2 = \cos(\beta - r) \cdot L'$$

La distancia total entre las filas de módulos será:

$$D = D1 + D2 = L \cdot \cos(Z) + \frac{[L \cdot sen\,(Z) - h2] \cdot \cos\,(\beta - r)}{\tan \alpha}$$

Para obtener la altura solar (α) y el acimut solar (β), se usarán las fórmulas:

$$sen\,(\alpha) = sen\,(\varphi) \cdot sen(\delta) + \cos(\varphi) \cdot \cos(\delta) \cdot \cos\,(\omega)$$

$$\cos(\beta) = \frac{sen(\varphi) \cdot sen(\alpha) - sen(\delta)}{\cos\,(\alpha) \cdot \cos\,(\varphi)}$$

siendo φ la latitud de la localización, δ la declinación solar y ω el ángulo horario.

La coordenada de la longitud de la localización no influye en los cálculos ya que en todas las localizaciones con igual latitud las distancias obtenidas serán las mismas.

La latitud se dará en grados. La declinación solar se calcula con la fórmula

$$\delta = 23.5 \sin(360 \frac{284+N}{365}),$$

con N el número de días transcurridos desde el 1 de enero. Se tomará para los cálculos el caso más desfavorable; δ del solsticio de invierno, -23.45º.

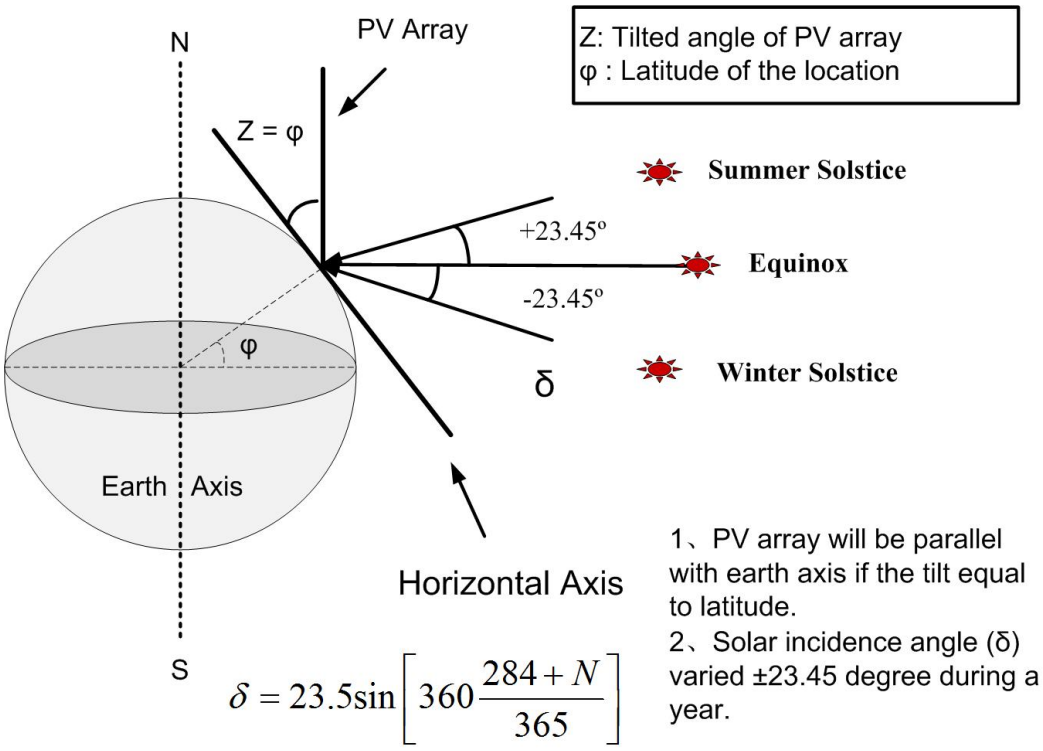

Figura 4.4. Localización de la fila de módulos y valores valor de declinación a lo largo del año.

El porcentaje del día sin sombra (con día se hace referencia a las horas que transcurren desde el amanecer hasta el atardecer) se aplicará sobre el ángulo horario de la salida y puesta del sol, para el cálculo del ángulo horario del periodo sin sombra. El ángulo horario se calcula con la fórmula:

$$\omega_s = \text{acos}[(-\tan(\delta)) \cdot \tan(\varphi)]$$

En la siguiente tabla se muestran diversos periodos sin sombra para diferentes latitudes:

Latitud φ (º)	Declinación δ(º)	Ángulo horario de la salida y la puesta del Sol ω_s	Ángulo horario del periodo sin sombra: 75% de la duración del día
0	-23.45	90,000	67,500
10	-23.45	85,613	64,210
20	-23.45	80,916	60,687
30	-23.45	75,496	56,591
36,25	-23.45	71,455	53,591
40	-23.45	68,655	51,491
50	-23.45	58,872	44,154
60	-23.45	41,295	30,971

Tabla 4.1. Condición de contorno de periodo sin sombra.

5 IMPLEMENTACIÓN INFORMÁTICA

En este capítulo se detallará la implementación de la herramienta informática para automatizar y facilitar el cálculo de la superficie requerida en una superficie fotovoltaica.

La herramienta consta de un 'script' ejecutable, denominado "herramienta.py", escrito en lenguaje Python, una carpeta con las librerías usadas y ejecutables para Microsoft Windows y Apple Mac OS. Este acepta como datos de entrada: la latitud del terreno, el porcentaje de día sin sombra, el ángulo de inclinación de los subconjuntos, orientación de los subconjuntos respecto al sur (azimut), diferencia de altura del terreno, ancho y largo del subconjunto de módulos. A su vez, la herramienta ofrece como resultado a una entrada de datos la distancia total entre subconjuntos de módulos y área ocupada por un subconjunto.

Puesto que los datos de entrada son para todos los casos de uso los mismos, se detallan a continuación los detalles necesarios par la correcta introducción de los datos:

Latitud del terreno:
Si se quiere obtener el cálculo para una latitud del hemisferio norte, se indicará en grados, en notación decimal. Por ejemplo, si se quiere calcular la latitud para 12º15'23'' en el hemisferio Norte (notación sexagesimal), se deberá indicar en la herramienta como 12.23639º, que es el equivalente en notación hexadecimal. Los decimales se indicarán con un punto (.). Los cálculos serán válidos para el hemisferio sur, y se pueden introducir valores negativos, pero los resultados serán idénticos a su correspondiente latitud en el hemisferio norte.

Porcentaje del día sin sombra:
Se introducirán los valores del porcentaje del día, es decir, el porcentaje de las horas de sol durante las cuales no habrá sombras mutuas en ningún punto del subconjunto posterior producida por el anterior. Los valores más habituales rondarían entre 60% y los 80%.

Ángulo de inclinación:
Se indicará en grados hexadecimales el ángulo de inclinación de los paneles. En muchos de los casos la inclinación se establece con valor igual a la latitud, pero se pueden modificar estos valores en la herramienta.

Orientación de los módulos con respecto al sur:
Para introducir la orientación en la herramienta, se indicará en grados. Los valores de la distancia entre subconjuntos y de área ocupada serán idénticos en casos de orientación Sur-Este o Sur-Oeste.

Diferencia de altura del terreno:
En los casos en los que los paneles estén colocados en suelos no nivelados. Si se trata de un suelo plano, se introducirá el valor "0". La distancia que se introduce se corresponde en el código con la variable *h2*, que es la distancia vertical medida entre las bases de los conjuntos, en metros.

Ancho del subconjunto de módulos:
Se introducirán los metros que ocupa un subconjunto a lo ancho, de manera que es necesario conocer las dimensiones del subconjunto completo para usar la herramienta. La unidad en este caso son metros, y se separarán los valores decimales con un punto (.)

Largo del subconjunto de módulos:
Se introducirán los metros que ocupa un subconjunto a lo largo, de manera que es necesario conocer las dimensiones del subconjunto completo para usar la herramienta. La unidad en este caso son metros, y se separarán los valores decimales con un punto (.)

Distancia sin sombra: Distancia mínima de separación entre apoyos para la que el subconjunto que se encuentra detrás no experimente sombra en ningún producida por el subconjunto situado delante suya en ningún punto.

Área que ocupa un subconjunto de paneles: Producto de la distancia sin sombra y el ancho de un subconjunto. Esta será el área máxima de sombra generada por el subconjunto. Si se coloca algún elemento dentro de este área se producirán sombras en el mismo.

Puesto que el propósito de una implementación informática es el hacer útil las conclusiones expuestas en anteriores secciones, se ha optado por una serie de criterios de diseño software sobre los que construir la herramienta.

5.1. Flexibilidad de uso

Un criterio básico ha sido el de la flexibilidad de uso. Puesto que esta herramienta puede ser usada por si sola o como parte de otro software se ha hecho hincapié en una interfaz flexible, de forma que la herramienta puede obtener sus datos de entrada y ofrecer sus datos de salida tanto por entrada estándar, típicamente un terminal, como vía ficheros o interfaz gráfica de usuario.

5.1.1 Desde el intérprete de comandos

Esta herramienta hace uso de la salida estándar para proporcionar facilidades de uso que buscan su funcionamiento en múltiples situaciones, con el fin de servir en diferentes escenarios en los que el cálculo de la superficie necesaria de una planta fotovoltaica entre en juego.

A continuación, se numerarán las distintas formas de uso de la herramienta desde el intérprete de comandos y posibles escenarios para dichos usos.

5.1.1.1 Ejecución en el intérprete de comandos

En esta sección nos referiremos al archivo ejecutable de nuestro programa como **herramienta.py** y asumiremos que todas las operaciones ejecutadas serán efectuadas en el

mismo directorio en el que nuestro programa reside. El separador decimal de los datos de entrada y salida numéricos será un punto.

Esta herramienta puede ser invocada desde el intérprete de comandos de un terminal. El programa puede así recibir su entrada como argumentos y escribir su salida en la salida estándar.

Como ejemplo, en caso de querer obtener los resultados para unos datos de entrada concretos se ofrece la opción de ejecutar el 'script' con los datos como argumentos de entrada.

Argumentos de entrada:
latitud de la localización (grados) = 35
porcentaje de día sin sombra (%) = 75
el ángulo de inclinación de los subconjuntos (grados) = 36.25
orientación respecto al sur (azimut) (grados) = 10
diferencia de altura del terreno (metros) = 0
ancho del subconjunto (metros) = 37.07
largo del subconjunto (metros) = 3.988

```
./herramienta.py –latitud 36.25 –porcentaje-sombra 75 –angulo-inclinacion 36.25 \
--orientacion h 10 –diferencia-altura 0 –ancho-subconjunto 37.07 –alto-subconjunto
3.988\ --detalle
```

Salida del programa:

```
Distancia entre subconjuntos: 12.025 Area ocupada por un subconjunto: 445.759
```

El programa ofrece los siguientes argumentos de entrada:

Nombre de argumento	Nombre abreviado	Referencia	Unidad	Ejemplo
latitud	p	Latitud del terreno	grados	--latitud 33.2
porcentaje-sombra	s	Porcentaje de día sin sombra	%	--porcentaje-sombra 72.1
angulo-inclinacion	Z	Ángulo de inclinación de los subconjuntos	grados	--angulo-inclinacion 37.8
orientacion	r	Orientación respecto al sur	grados	--orientacion 10
diferencia-altura	hd	Diferencia de altura del terreno	metros	--diferencia-altura 0
ancho-subconjunto	K	Ancho del subconjunto	metros	--ancho-subconjunto 37.07
alto-subconjunto	L	Alto del subconjunto	metros	--alto-subconjunto 2.734
detalle	-v	Salida del programa más detallada	-	--detalle
fichero	-f	Ruta de archivo de entrada de datos		--fichero "entrada.txt"

Cada argumento puede ser indicado en su forma expandida o en su forma abreviada. El resultado es el mismo en ambos casos.

```
Forma expandida del argumento porcentaje-sombra
./herramienta.py –porcentaje-sombra 75
Forma abreviada del argumento porcentaje-sombra
./herramienta.py -s 75
```

El programa ofrece los siguientes datos de salida:

Datos de salidas:	**Ejemplos:**
Distancia entre subconjuntos	32.112
Area ocupada por un subconjunto	532.09

5.1.1.2 Ejecución a través del intérprete de comandos

El uso desde el intérprete de comandos es realmente conveniente para la incorporación de la herramienta como parte de una cadena de instrucciones mayor. De esta forma, el uso de esta herramienta puede convertirse en parte de una serie de cálculos más complejos, puede ser un complemento a otras operaciones que se quieran efectuar. Puesto que dichas operaciones ajenas a nuestra herramienta pueden ser programadas en un lenguaje y contexto de programación distinto a Python, el hecho de que esta herramienta haga uso de la entrada y salida estándar proporciona una interfaz entre entornos que permite su uso de forma extendida sin ser el lenguaje de programación o el sistema operativo[1] una barrera.

La estimación de las dimensiones necesarias para la instalación de un complejo industrial que incluya una planta fotovoltaica puede servir a modo de ejemplo. Puesto que parte de esta operación incluye el hallar la superficie necesaria para la instalación fotovoltaica con unos parámetros de entrada que definan nuestros requisitos, podría efectuarse el cálculo de dicha superficie mediante la llamada a nuestra herramienta como un procedimiento externo o como parte de una cadena de comandos (“piping”).

5.1.1.3 Ficheros como interfaz

Una de las opciones proporcionadas por la herramienta es el uso de ficheros como interfaz. Los datos de entrada pueden ser provistos desde un archivo de entrada y los datos de salida pueden ser obtenidos desde un archivo de salida.

La herramienta espera un cierto formato del fichero de entrada. Cada línea ha de contener los argumentos de entrada de una llamada concreta a nuestra herramienta presentados sin espacios y separados por una coma. Cada entrada estará diferenciada de otra por un salto de línea.

La salida será un fichero de nombre “salida.txt” que presentará los datos de salida línea a línea correspondientes a los datos de entrada.

[1] A fecha de publicación, tanto sistemas operativos de la familia Unix como los Microsoft Windows más recientes proveen de una entrada y salida estándar ampliamente compatible. Microsoft Windows ofrece PowerShell, una herramienta que proporciona alternativa en alto grado compatible con los sistemas Unix.

Ejemplo de contenido fichero entrada:

```
35.3,70,45,10,0,34,3
35.3,70,35,10,0,34,3.1
35.3,78,37,10,0,34,3.2
35.3,71,35,10,0,34,3.3
35.3,79,35,10,0,34,3.3
35.3,72,35,10,0,34,3.3
35.3,78,35,10,0,34,3.1
```

Ejemplo de contenido fichero salida:

```
8.79,298.875
8.129,276.4
10.241,348.184
8.805,299.383
10.544,358.481
8.968,304.907
9.635,327.588
```

Una vez creado el fichero de entrada puede obtenerse la salida haciendo uso del argumento –fichero.

Ejemplo de uso de la herramienta con fichero entrada como argumento:

```
./herramienta.py --fichero "entrada.txt"
```

Esta opción puede ser interesante para hacer cálculos en conjuntos grandes de entradas y sacar conclusiones, como en el capítulo 7 de este documento.

5.1.2 Interfaz gráfica de usuario

Si se ejecuta el 'script' sin argumentos, aparecerá la interfaz gráfica de usuario.

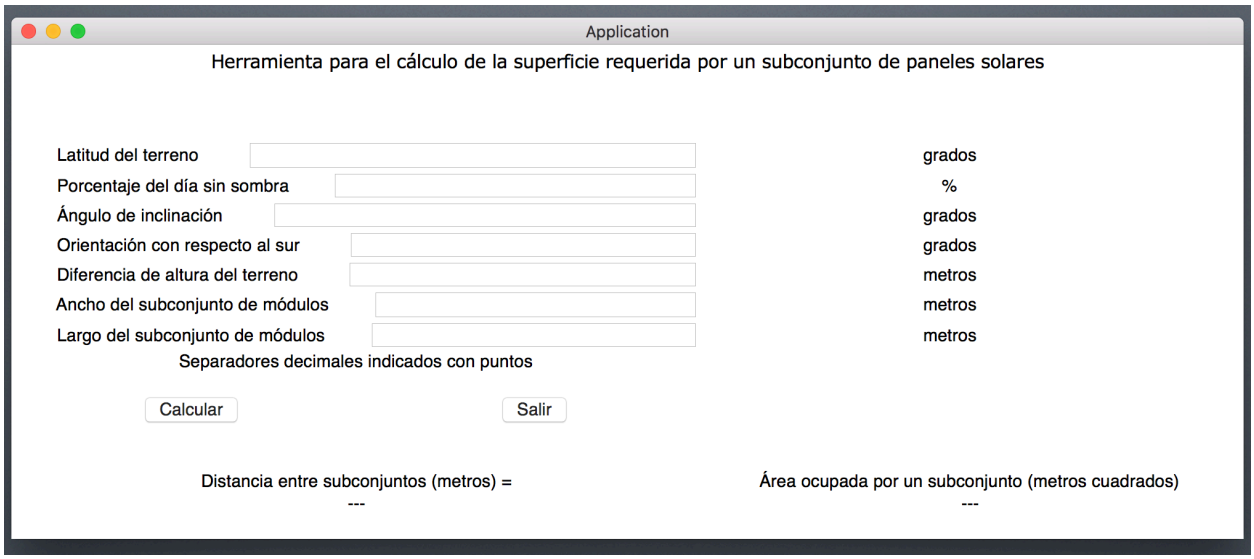

Figura 5.1. Interfaz gráfica sin datos insertados

Al introducir los distintos datos y accionar el botón "Calcular" podrán obtenerse los resultados. El botón "Salir" conduce a la salida de la herramienta.

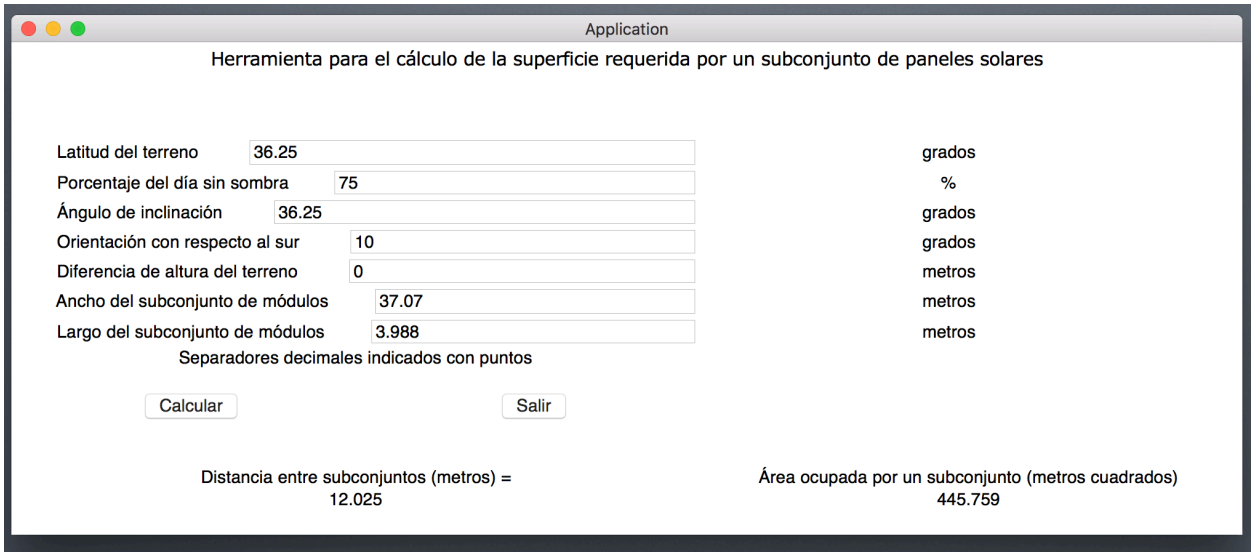

Figura 5.2. Interfaz gráfica tras realizar un cálculo

La herramienta provee interfaz gráfica tanto para Microsoft Windows como para Mac OS.

5.2. Accesibilidad para el desarrollador

Otro criterio que se ha tenido en cuenta es el de la accesibilidad. Por accesibilidad nos referimos a la facilidad de comprensión y edición del programa. Un caso representativo de la importancia de la accesibilidad sería el de una persona que quisiese extender la funcionalidad de esta herramienta, tanto la comprensión del código de programación de la herramienta como la facilidad para importar módulos externos o incluir nuevo código.

Por la facilidad de uso, potencia, portabilidad, cantidad de recursos disponibles, coste, enfoque generalista y popularidad entre los desarrolladores se ha escogido el lenguaje de programación Python[2]. Python es un lenguaje de programación muy potente, de uso extendido en todo tipo de industrias y comunidad de desarrolladores. La cantidad de módulos disponibles es muy grande en comparación con otros lenguajes potentes y populares, y su comprensión puede ser fácil para personas familiarizadas con lenguajes de programación. En la elección de Python ha pesado especialmente el hecho de que sea un lenguaje interpretado, lo que permite la ejecución de código escrito en Python en la práctica totalidad de entornos informáticos. Así, esta herramienta podrá ser usada en máquinas que corran un sistema tipo Microsoft Windows o Mac OS entre otros. En un principio se planteó la existencia de una interfaz web para la herramienta, opción especialmente factible debido a que Python cuenta con múltiples opciones para el desarrollo web. Finalmente, esta opción quedó descartada.

[2] La versión de Python usada para la herramienta es la 2.7.14. El ecosistema de Python está en el momento de publicación de este documento en transición hacia la versión 3.x. No obstante, el portar el código a la versión 3.x es una operación trivial.

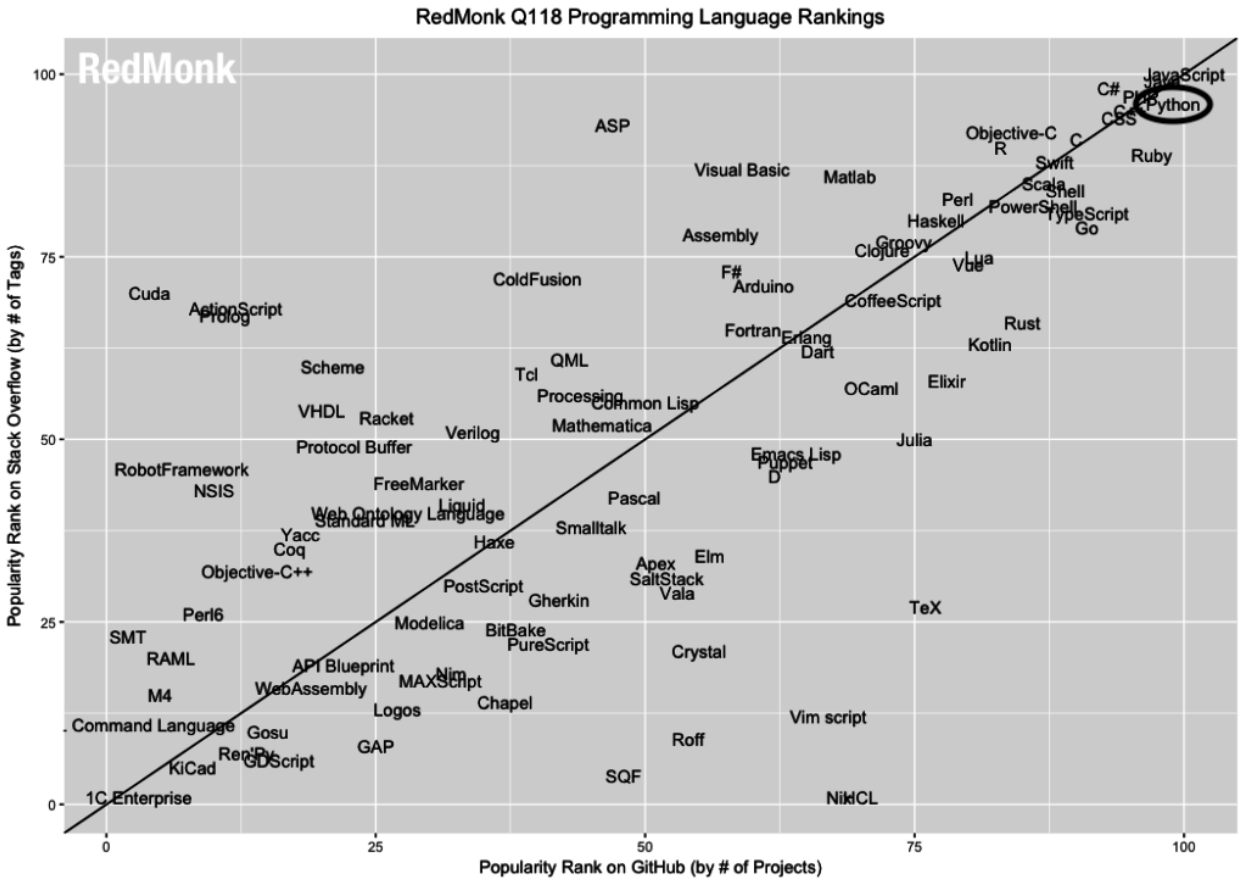

Figura 5.3. Ranking de popularidad de lenguajes de programación RedMonk. (Enero 2018)

En un principio se planteó para el desarrollo de esta herramienta el entorno Matlab, debido a su uso durante el estudio de esta licenciatura, pero Python aporta mayor accesibilidad y recursos suficientes para la clase de operaciones matemáticas requeridas.

En caso de querer editar la herramienta sólo se necesitaría abrir el archivo *herramienta.py* y editar el código.

Extender la funcionalidad es cuestión de incluir código propio o de terceros en la herramienta. Para el uso de código de terceros Python incorpora un sistema de módulos (librerías de programación) especialmente potente y sencillo de usar. Además, existen grandes repositorios de módulos para Python en Internet y programas que facilitan la instalación de dichos módulos.

Para este proyecto el proceso de incorporación de módulos ha sido el de la realización de una búsqueda en los repositorios de módulos de los principales módulos proveedores de la funcionalidad requerida, la posterior comparación de los más robustamente soportados por sus desarrolladores y finalmente la incorporación del módulo mejor valorado por medio de su descarga e importación en el código de la herramienta.

A modo de ejemplo podemos tratar el caso de la incorporación del módulo para la salida por interfaz gráfica de usuario. El objetivo era el de proveer de una interfaz gráfica de usuario funcional, sencilla y que no entrañase gran complejidad en su uso a nivel de implementación. Evitando la complejidad evitaríamos dificultades en el desarrollo de la herramienta para este proyecto y a futuros editores de la misma. Tras una búsqueda en los principales repositorios de módulos se decidió entre cuatro diferentes módulos capaces de proveer de una interfaz gráfica en

las principales plataformas (Microsoft Windows, Linux, Mac OS). La mayor sencillez de "appJar" para la construcción de la interfaz lo hizo óptimo para este propósito. La instalación del módulo se efectuó con el gestor de paquetes Pip. En la línea de comandos se invocó a este con la instrucción "pip install appjar" y Pip descargó el modulo y lo distribuyó en el entorno de desarrollo al directorio desde el cual los módulos de Python son ejecutados.

Ejemplo de instalación del módulo appJar para la herramienta:

```
pip install appjar
```

Ejemplo importación y uso del módulo appJar en herramienta.py:

```
from appJar import gui
app = gui("Aplicacion", "1000x400")
…
```

Por último, cabe destacar el hecho de que Python y la inmensa mayoría de módulos para este son de código abierto. Esto facilita tremendamente el uso, manipulación y distribución de la herramienta.

5.3. Código

A continuación, se proporciona el código de implementación y correspondiente explicación. Algunas líneas han sido omitidas para evitar plagio. También se puede observar que parte de comentarios del código están en inglés, esto es por mantener la coherencia con el entorno, estando ya mucho de lo provisto por Python y sus módulos en inglés se ha determinado conveniente mantener ciertas partes en este idioma.

```
#!/usr/bin/python   # inform the shell what interpreter it should use
# encoding should be specified for use in windows OS
# -*- coding: utf-8 -*-

# This script can input/output data from/to the standard input/output,
# via a input/output files or by using a graphical user interface

import math as m
import sys
import argparse

reload(sys)  # Reload to get sys.setdefaultencoding()
# from now on this document will be read as UTF8 encoded
sys.setdefaultencoding('UTF8')
```

```
# Principal function -----------------------
# Definitions for trig functions using degrees.
def dsin(a):
    return m.sin(m.radians(a))
def dcos(a):
    return m.cos(m.radians(a))
def dtan(a):
    return m.tan(m.radians(a))
def dasin(a):
    return m.degrees(m.asin(a))
...
def datan(a):
    return m.degrees(m.atan(a))

# Returns two floats representing total distance between arrays and
# total area per array
def herramienta(phi, shadow_percentage, Z, r, h2, K, L):
    # Location data, in degrees
    # SOLAR DECLINATION in WINTER, in degrees. -23.45 on winter solstice.
    # (23º 26')(condition of longest N S shading)
    delta = -23.45
    # omega represents the hour angle of the day
    omega =  dacos((-dtan(delta)) * dtan(phi))
    # omegaset represents the shadow percentage applied to the hour angle
    omegaset = float(shadow_percentage)/100 * omega
    # Data from PV array
    H =  L * dsin(Z)    # H == Height of front PV arrays, in meters
    h1 = H - h2    # h1 = relative height between arrays
    # CALCULATIONS
    # solar calculations for beta and alpha. Angles in degrees
    # alpha == solar altitude
    alpha = dasin(dsin(phi) * dsin(delta) + dcos(phi) *
             dcos(delta) * dcos(omegaset))
    # beta == solar azimuth.
    beta = dasin( dcos(delta) * dsin(omegaset) / dcos(alpha) )
    L1 = h1 / dtan(alpha)   # L1 == shading distance behind PV arrays
    # Calculation of geometrics of arrays, distances in meters
    # D1== Horizontal distance between bottom and top of the same array
    D1 = L * dcos (Z)
    # D2 == Horizontal projection from top to the array to bottom
    # to the next one.
    # R is the array azimuth
    D2 = dcos (beta-r) *  L1

    D = D1 + D2  # Total distance between arrays
    area_array = K * D  # Total area per array
```

```
    roundedDistance = round(D, 3)
    roundedArea = round(area_array, 3)


    return [roundedDistance, roundedArea]
# ---------------------- END Principal function


# Standard I/O handling ----------------------
parser = argparse.ArgumentParser() # Instantiate a parser
# Define arguments that will be accepted
# In case the user wants to read data from a file,
# this -f/--fichero flag(argument) will
# allow the user to enter the path to the file.
# Can be just the name if the file is in the same directory he/she is running
# this script from
parser.add_argument("-f","--fichero",
                    help="""Introduce la ruta del fichero de entrada. En el fichero
cada línea ha de contener los valores de los campos de entrada separados por una coma
('35.3,70,35,10,0,34,3.1').Ejemplo:'entrada.txt'""")
# Help argument is for showing that text argument as inline help text when
# the user runs our script with -h(also --help) flag as argument
parser.add_argument("-p", "--latitud", help="Latitud del terreno. Ejemplo:35.3")
parser.add_argument("-s", "—porcentaje-sombra",
                     help="Porcentaje del dia de sombra. Ejemplo:71")
parser.add_argument("-Z", "—angulo-inclinacion",
                     help="Angulo de inclinacion del subconjunto")
parser.add_argument("-r","--orientacion", help="Orientacion del subconjunto")
parser.add_argument("-hd","—diferencia-altura",
                     help="Diferencia de Altura del terreno")
parser.add_argument("-K","--ancho-subconjunto", help="Ancho del subconjunto")
parser.add_argument("-L","—alto-subconjunto", help="Alto del subconjunto")
parser.add_argument("-v","--detalle",
                    help=" Imprime un resultado detallado. No recibe valor. Ejemplo:
'-v'",
                    action='store_true')
# Get all the arguments entered by user and store them in args instance
args = parser.parse_args()

has_args = False    # Store knowledge on having args entered by user or not
# Check if args has been entered
for arg in vars(args): # vars takes a class instance and returns a __dict__
    if getattr(args, arg) is not None and getattr(args, arg) is not False:
        has_args = True
        break

# Case the user enters arguments
```

```
if has_args:
    # user (by using the argument -f) pass an entry file path. If -f/-fichero
    # argument has been entered, forget about the rest of arguments
    if args.fichero is not None:
        # Open files
        file = open(args.fichero, "r")
        out_file = open("salida.txt", "write")
        if file:
            for entry in file:
                # Remove trailing whitespace (like \n)
                stripped_entry = entry.rstrip()
                # split data in units sparated by ','
                splitted_entry = stripped_entry.split(",")
                # convert string data to float
                parsed_entry = map(float, splitted_entry)
                # call herramienta function with data
                result = map(str, herramienta(*parsed_entry))
                # make the result an string with a new line
                # indication at the end
                parsed_result = ",".join(result) + "\n"
                # write the result on the output file
                out_file.write(parsed_result)
        # Close files
        file.close()
        out_file.close()
    # User enters data directly from standard I/O
    # If -f/-fichero argument has not been entered, ignore it
    elif (args.latitud and args.porcentaje-sombra and args.angulo-inclinacion
            and args.orientacion and  args.diferencia-altura and
             args.ancho-subconjunto and args.alto-subconjunto):
        entry = [args.latitud, args.porcentaje-sombra, args.angulo-inclinacion,
                args.orientacion,  args.diferencia-altura,
                 args.ancho-subconjunto, args.alto-subconjunto]
        parsedEntry = map(float, entry)  # make all arguments float
        result = herramienta(*parsedEntry)
        # Print result
        # If detalle argument has been entered show detailed output
        if args.detalle:
            print "Distancia entre subconjuntos: " + str(result[0]),
                    "Area ocupada por un subconjunto: " + str(result[1])
        else:   # no detalle argument => simple output
            print result[0], result[1]
    else:   # In case any required argument is missing
        # Get a list of arguments that are different from None or 'fichero'
        missing_args = filter(lambda arg: getattr(args, arg) is None
                                            and arg != 'fichero', vars(args))
        print "Los siguientes campos son requeridos: " + ", ".join(missing_args)
```

```
# ---------------------- END standard I/O handling



# graphical User Interface ----------------------
if (not has_args):  # User does not provide args
    # Import GUI appJar library
    from appJar import gui

    # create a GUI instance called app
    app = gui("Application", "1000x400")
    app.setBg("white")
    app.setFont(14)

    # add widgets
    app.addLabel("title", "Herramienta para el cálculo de la superficie
                  requerida por un subconjunto de paneles solares",0,0,5)
    app.getLabelWidget("title").config(font=("Verdana","15","normal"))
    # entries with empty strings are used to make space in the UI

...
...
    app.addLabelNumericEntry("Latitud del terreno",4,0)
    app.addLabel("l1","grados",4,1)
    app.addLabelNumericEntry("Porcentaje del día sin sombra",5,0)
    app.addLabel("l2","%        ",5,1)
    app.addLabelNumericEntry("Ángulo de inclinación",6,0)
    app.addLabel("l3","grados",6,1)
    app.addLabelNumericEntry("Orientación con respecto al sur",7,0)
    app.addLabel("l4","grados",7,1)
    app.addLabelNumericEntry("Diferencia de altura del terreno",8,0)
    app.addLabel("l5","metros",8,1)
    app.addLabelNumericEntry("Ancho del subconjunto de módulos",9,0)
    app.addLabel("l6","metros",9,1)
    app.addLabelNumericEntry("Largo del subconjunto de módulos",10,0)
    app.addLabel("l7","metros",10,1)
    app.addLabel("l8", "Separadores decimales indicados con puntos")
    app.getLabelWidget("l8").config(font="Helvetica 14 normal")
    app.addLabel("   ") # UI spacing
    # Function to process entered data in the UI after a click
    # in "Calcular" or "Cancelar" is performed
    def on_click(button):
        if button == "Cancelar":
            app.stop()
        else:
            # Get values from input fields in the UI
            phi = app.getEntry("Latitud del terreno")
```

```
            shadow_percentage = app.getEntry("Porcentaje del día sin sombra")
            Z = app.getEntry("Ángulo de inclinación")
            r = app.getEntry("Orientación con respecto al sur")
            h2 = app.getEntry("Diferencia de altura del terreno")
            K = app.getEntry("Ancho del subconjunto de módulos")
            L = app.getEntry("Largo del subconjunto de módulos")
            # Some values need to be absolute value
            result = herramienta(abs(phi), shadow_percentage,
                                 abs(Z), abs(r), h2, K, L)
            distancia = result[0]
            area = result[1]
            # Set result fields in the UI with the result of calculation
            app.setLabel("lresult0", distancia)
            app.setLabel("lresult1", area)


    # link the buttons to the function called on_click
    app.addButtons(["Calcular", "Salir"], on_click)
    app.addLabel("    ")    # UI spacing
    app.addLabel("     ")    # UI spacing
    app.addLabel("ltextresult0", "Distancia entre subconjuntos (metros) =")
    app.addLabel("lresult0", "---")
    app.addLabel("       ")    # UI spacing
    app.addLabel("ltextresult1", "Área ocupada por un subconjunto (metros
                                  cuadrados)",16,1)
    app.addLabel("lresult1", "---",17,1)
    app.addLabel("        ")    # UI spacing

    # start the GUI
    app.go()

# ---------------------- END graphical user interface
```

5.3.1. Funciones trigonométricas

El código necesitará implementar funciones trigonométricas; seno, coseno, tangente, arcoseno, arcoseno y arcotangente. Las funciones trigonomométricas es necesario importarlas antes de poder hacer uso de ellas, ya que no forman parte de las funciones básicas de Python. Para importar la librería matemática que incluye las funciones que usaremos es necesario escribir al inicio del código la instrucción:

```
import math
```

Esta línea de código hará que se incluya toda la librería de funciones matemáticas (math en inglés), y para usar cualquiera de estas funciones se deberá escribir el prefijo “math.” y a

continuación la función que queramos usar. Por ejemplo, para usar la función seno de phi, se escribiría lo siguiente:

```
math.sin(phi)
```

En cambio, a modo de alias, se puede otorgar otro nombre al elemento importado:

```
import math as m
```

al usar esta librería, deberemos, a partir de ahora, invocarla como "m". Añadiendo un punto tras *m* podremos hacer uso de las funciones que la librería "math" proporciona.

```
m.sin(phi)
```

lo cual reduce ligeramente la nomenclatura.

Las funciones trigonométricas que vamos a usar en Python están definidas para trabajar con radianes y no con grados, de manera que para usarlas correctamente habría que realizar el cambio a grados para introducir el valor del ángulo. En cambio, para agilizar la nomenclatura y facilitar la lectura del código se han creado lo que se definen como funciones "wrapper" (o envoltorio en inglés), cuyo propósito principal es tomar la función original y realizar un pequeño cambio para adaptarla a nuestra intención. Sin esto, el código sería más difícil de leer y menos intuitivo.

Crearemos por tanto una función para cada una de las funciones que deseamos utilizar: el seno, coseno, tangente y sus funciones inversas. Usaremos además las funciones *degree* y *radians*:
Degrees es una función que tiene como entrada un ángulo en radianes y devuelve la conversión de este ángulo en grados.
Radians es una función que tiene como entrada un ángulo en grados y devuelve la conversión de este ángulo en radianes.
Ambas funciones pertenecen a la librería "*math*", de manera que necesitarán el prefijo indicado anteriormente para poder usarse.

El cálculo del seno de cualquier ángulo en grados, queda:

```
def dsin(a):
    return m.sin(m.radians(a))
```

Donde la función, que siempre se define usando el término *def*, seguido del nombre que se elige la función y posteriormente el argumento entre paréntesis seguido de dos puntos.
En este caso se ha tomado como nombre para esta función el término seno (*sin* en inglés) precedido de la letra d (de *degrees* en inglés).
A continuación, tras la indentación, se escribe el término *return*, indicativo de que la función devolverá el resultado del código que está escrito a continuación de dicho término. En este caso, se devolverá el resultado de tomar el ángulo *a*, inicialmente en grados, se pasará a radianes, y posteriormente se calculará el seno. El resultado de estos cálculos se devolverá como resultado de la función *dsin*. Así, esta será una función "envoltorio" de la función *sin*.

Para las funciones arcotangente, arcoseno y arcoseno, se hará uso de la función *degree*, ya que el resultado de aplicar el arcoseno será radianes, y se necesitará en grados. Por ejemplo, la función "wrapper" de arcoseno:

```
def dasin(a):
    return m.degrees(m.asin(a))
```

Se definirá de nuevo con el prefijo d, de *degrees*, por el uso de esta función en grados y no en radianes. La entrada de esta función será el valor *a*, al que se le aplica el arcoseno y posteriormente se convierte el resultado a grados, ya que se encontrará por defecto en radianes. El resultado de estos cálculos será la salida de la función *dasin*.

Aplicando este concepto a cada función trigonométrica se obtiene el siguiente segmento de código:

```
def dsin(a):
    return m.sin(m.radians(a))
def dcos(a):
    return m.cos(m.radians(a))
def dtan(a):
    return m.tan(m.radians(a))
def dasin(a):
    return m.degrees(m.asin(a))
…
def datan(a):
    return m.degrees(m.atan(a))
```

5.3.2. Función Principal

A continuación, en el código se introduce la función que calcula la distancia entre subconjuntos y el área ocupada por subconjunto, acorde con la geometría del modelo, explicado con detalle en el capítulo anterior, y con la nomenclatura acorde con la figura 4.3.

La función principal recibe como argumentos la latitud de la localización, phi, el porcentaje de sombra, shadow_percentage, el ángulo de inclinación de los módulos, Z, la diferencia de alturas del terreno, h2. La variable K será el ancho del subconjunto de módulos y L el largo.

```
def herramienta(phi, shadow_percentage, Z, r, h2, K, L):
    # Location data, in degrees
    # SOLAR DECLINATION in WINTER, in degrees. -23.45 on winter solstice.
    # (23º 26')(condition of longest N S shading)
    delta = -23.45
```

Se calcula el ángulo solar, sobre el que se aplica el porcentaje de sombra deseado.

```
    # omega represents the hour angle of the day
    omega =  dacos((-dtan(delta)) * dtan(phi))
    # omegaset represents the shadow percentage applied to the hour angle
    omegaset = float(shadow_percentage)/100 * omega
```

```
# Data from PV array
H =  L * dsin(Z)    # H == Height of front PV arrays, in meters
h1 = H - h2    # h1 = relative height between arrays
```

Cálculo de la altura solar y el acimut solar:

```
# CALCULATIONS
# solar calculations for beta and alpha. Angles in degrees
# alpha == solar altitude
alpha = dasin(dsin(phi) * dsin(delta) + dcos(phi) *
         dcos(delta) * dcos(omegaset))
# beta == solar azimuth.
beta = dasin( dcos(delta) * dsin(omegaset) / dcos(alpha) )
```

Cálculo de la proyección horizontal de los rayos solares entre dos filas de subconjuntos:

```
L1 = h1 / dtan(alpha)   # L1 == shading distance behind PV arrays
# Calculation of geometrics of arrays, distances in meters
```

Cálculo de la distancia ocupada por un subconjunto y de la sombra máxima producida.

```
# D1== Horizontal distance between bottom and top of the same array
D1 = L * dcos (Z)
# D2 == Horizontal projection from top to the array to bottom
# to the next one.
# R is the array azimuth
D2 = dcos (beta-r) *  L1
```

Suma de las distancias para calcular la distancia total, D

```
D = D1 + D2  # Total distance between arrays
area_array = K * D  # Total area per array
```

Se aplica un redondeo a tres cifras antes de mostrar el cálculo.

```
roundedDistance = round(D, 3)
roundedArea = round(area_array, 3)
```

La función devuelve como salidas la distancia entre subconjuntos y el área por subconjunto.

```
return [roundedDistance, roundedArea]
```

5.3.3. Interfaces

Como se ha mencionado con anterioridad, hay tres formas principales de interactuar con la aplicación; tres variantes de interfaz que se implementan tras la función principal.

Dos de las variantes de interfaz hacen uso de la entrada estándar, recibiendo argumentos al invocar a esta herramienta.

Ejemplo uso de herramienta mediante la línea de comandos

```
./herramienta.py --fichero "entrada.txt"

./herramienta.py –latitud 36.25 –porcentaje-sombra 75 –angulo-inclinacion 36.25 \
--orientacion h 10 –diferencia-altura 0 –ancho-subconjunto 37.07 –alto-subconjunto
3.988\ --detalle
```

Para poder procesar los argumentos que la herramienta recibe, Python provee un módulo con este propósito en su librería estándar: "argparse".

Haremos uso de "argparse" para capturar y adaptar los datos recibidos como argumentos a conveniencia.

El proceso de captura y adaptación empieza con la importación de argparse.

Ejemplo uso de herramienta

```
import argparse
```

Obtención de una instancia de ArgumentParser

```
parser = argparse.ArgumentParser()
```

Mediante 'argparser' se indican los posibles argumentos que se deben capturar. Por ejemplo, aquellos argumentos precedidos de "-p" o "--latitud". También se indica qué texto de ayuda mostrar cuando alguien invoca la herramienta con el argumento "-h" o "--help". Este es un argumento especial, ofrecido por las herramientas de línea de comandos que argparse reserva para tal propósito. Al invocar a la herramienta con el argumento "help", argparse mostrará todos los posibles argumentos y la explicación de ayuda provista.

Parseado de argumentos

```
…
parser.add_argument("-p", "--latitud", help="Latitud del terreano. Ejemplo:35.3")
…
parser.add_argument("-r","–orientacion", help=" Orientación con respecto al sur ")
 …
```

Ejemplo de la salida que obtenemos al invocar la herramienta con el argumento "–help"

```
usage: herramienta.py [-h] [-f FICHERO] [-p LATITUD] [-s PORCENTAJE_SOMBRA]
                      [-Z ANGULO_INCLINACION] [-r ORIENTACION]
                      [-hd DIFERENCIA_ALTURA] [-K ANCHO_SUBCONJUNTO]
                      [-L ALTO_SUBCONJUNTO] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -f FICHERO, --fichero FICHERO  Introduce la ruta del fichero de entrada. En el
                        fichero cada línea de ha de contener los valores de los campos
                        de entrada separados por una coma.'35.3,70,35,10,0,34,3.1'.
                        Ejemplo:'entrada.txt'
  -p LATITUD, --latitud LATITUD     Latitud del terreno. Ejemplo:35.3
  -s PORCENTAJE_SOMBRA, --porcentaje-sombra PORCENTAJE_SOMBRA
```

```
                        PORCENTAJE_SOMBRA. Ejemplo:71
  -Z ANGULO_INCLINACION, --angulo-inclinacion ANGULO_INCLINACION
                        Angulo de inclinacion del subconjunto.
  -r ORIENTACION, --orientacion ORIENTACION
                        Orientacion del subconjunto
  -hd DIFERENCIA_ALTURA, --diferencia-altura DIFERENCIA_ALTURA
                        Diferencia de altura del terreno
  -K ANCHO_SUBCONJUNTO, --ancho-subconjunto ANCHO_SUBCONJUNTO
                        Ancho del subconjunto
  -L ALTO_SUBCONJUNTO, --alto-subconjunto ALTO_SUBCONJUNTO
                        Alto del subconjunto
  -v, --detalle         Imprime un resultado detallado. No recibe valor. Ejemplo: '-v'
```

Una vez declarados todos los argumentos que prevé la herramienta, se indica su captura y almacenamiento en la variable "args".

```
args = parser.parse_args()
```

Para poder determinar qué está solicitando el usuario, se procede al análisis de los argumentos. En primer lugar se determina si se ha proporcionado algún argumento.

```
has_args = False    # Store knowledge on having args entered by user or not
# Check if args has been entered
for arg in vars(args): # vars takes a class instance and returns a __dict__
    if getattr(args, arg) is not None and getattr(args, arg) is not False:
        has_args = True
        break
```

La variable "has_args" será igual a el valor "True" o "False" y en función de este valor se bifurca el comportamiento de la herramienta, dando lugar, en caso de que esta variable sea igual a "True", a que se proceda al análisis de los argumentos, y en caso de que esta sea igual a "False" a que se proceda a mostrar la interfaz gráfica de usuario.

Si el argumento precedido de "-f" o "--fichero" ha sido proporcionado, se asume que se pretende leer la entrada de datos de un fichero, por lo que los demás argumentos serán ignorados.

En este punto se asume que la entrada debe ser de este tipo, con una ruta de fichero como entrada
./herramienta.py --fichero "entrada.txt"

```
# Case the user enters arguments
if has_args:
    # user (by using the argument -f) pass an entry file path. If -f/-fichero
argument has been entered, forget about the rest of arguments
    if args.fichero is not None:
```

Una vez comprobado el argumento "--fichero" o "-f" y capturado su valor se utiliza como la ruta a un fichero de entrada. Esta información permite abrir el fichero que se entenderá como entrada y también se creará un fichero de salida. Con los ficheros disponibles se procede a su lectura e interpretación del fichero de entrada. Este fichero debe tener un formato de manera que la herramienta pueda leerlo e interpretarlo, de ahí que en su interpretación haga ciertas comprobaciones y tratamiento de datos. Este procedimiento se efectúa línea a línea y el resultado se almacena como lista de datos de entrada. Esta lista es a su vez usada como entrada de la

función principal que devuelve un resultado que se almacena en la siguiente línea disponible del fichero de salida que hemos dispuesto con antelación.

```
# Open file
file = open(args.file, "r")
out_file = open("salida.txt", "write")
if file:
    for entry in file:
        # Remove trailing whitespace (like \n)
        stripped_entry = entry.rstrip()
        # split data in units sparated by ','
        splitted_entry = stripped_entry.split(",")
        # convert string data to float
        parsed_entry = map(float, splitted_entry)
        # call herramienta function with data
        result = map(str, herramienta(*parsed_entry))
        # make the result an string with a new line
        # indication at the end
        parsed_result = ",".join(result) + "\n"
        # write the result on the output file
        out_file.write(parsed_result)
# close file
file.close()
out_file.close()
```

Una vez terminada la escritura del resultado de procesar la última línea del fichero de entrada se cierran los ficheros de entrada y de salida y se termina el programa.

Si el argumento precedido de “-f” o “--fichero” no ha sido capturado, se procede a comprobar qué otros argumentos han sido capturados.

Ejemplo de tipo de entrada que se contempla en este punto del código

```
./herramienta.py –latitud 36.25 –porcentaje-sombra 75 –angulo-inclinacion 36.25 \
--orientacion h 10 –diferencia-altura 0 –ancho-subconjunto 37.07 –alto-subconjunto
3.988\ --detalle
```

```
# User enters data directly from standard I/O
# If -f/-fichero argument has not been entered, ignore it
elif (args.latitud and args.porcentaje-sombra and args.angulo-inclinacion
        and args.orientacion and  args.diferencia-altura and
         args.ancho-subconjunto and args.alto-subconjunto):
    entry = [args.latitud, args.porcentaje-sombra, args.angulo-inclinacion,
            args.orientacion,  args.diferencia-altura,
             args.ancho-subconjunto, args.alto-subconjunto]
    parsedEntry = map(float, entry)  # make all arguments float
    result = herramienta(*parsedEntry)
    # Print result
     # If detalle argument has been entered show detailed output
    if args.detalle:
```

```
            print "Distancia entre subconjuntos: " + str(result[0]),
                  "Area ocupada por un subconjunto: " + str(result[1])
        else:   # no detalle argument => simple output
            print result[0], result[1]
```

En caso de poderse verificar que todos los argumentos han sido proporcionados, estos se convierten de tipo “String” a tipo “Float” y se pasan como argumentos para la función principal cuyo resultado se imprime por la salida estándar.

Hay un argumento extra, “-v” o “--detalle” que permite especificar a la herramienta que imprima el resultado con algo más de detalle. El motivo de esta opción es poder elegir entre un resultado más adecuado para la lectura por parte de un ser humano o para la lectura del resultado por parte de otros programas.

Si alguno de los argumentos no ha sido proporcionado, se devuelve un aviso de ello como la salida de la herramienta, especificando qué argumento es requerido.

```
    else:   # In case any required argument is missing
        # Get a list of arguments thar are different from None or 'file'
        missing_args = filter(lambda arg: getattr(args, arg) is None
                                          and arg != 'fichero', vars(args))
        print "The following data is required: " + ", ".join(missing_args)
```

Ejemplo de salida adecuada a la lectura por parte de un ser humano

```
distance between panels: 8.129 total area per array: 276.4
```

Ejemplo de salida adecuada a la lectura por parte de otro programa

```
8.129 276.4
```

Con anterioridad se dejó pendiente el explicar con más detalle qué flujo de ejecución sigue la herramienta en caso de no recibir argumento alguno. Dado este caso la herramienta importa “appJar”, una librería para la creación de interfaces graficas de usuario. Esta se inicializa con las dimensiones que tendrá la ventana de la herramienta y con esta instancia de la aplicación se pasa a definir una serie de configuraciones gráficas y los campos que necesitamos que el usuario de la aplicación complete. Con estos definidos la herramienta inicia la interfaz gráfica con la instrucción ¨app.go()”.

```
    app = gui("Application", "1000x400")
    app.setBg("white")
    app.setFont(14)

    # add widgets
    app.addLabel("title", "Herramienta para el cálculo de la superficie
                  requerida por un subconjunto de paneles solares",0,0,5)
    app.getLabelWidget("title").config(font=("Verdana","15","normal"))
    # entries with empty strings are used to make space in the UI
    # (appJar seems to be buggy when label have long text)
...
...
    app.addLabelNumericEntry("Latitud del terreno",4,0)
...
```

```
# start the GUI
app.go()
```

En este momento el usuario presenciará la apertura de una ventana que muestra la interfaz gráfica de la herramienta. Una vez los datos están rellenos y el usuario pulse "Calcular" o "Cancelar" el programa llama a la función "on_click". En esta primero se capturan los datos introducidos por el usuario, después se procesan y se invoca a la función principal con estos datos como entrada. Una vez se ha obtenido el resultado se actualiza la interfaz gráfica con dicho resultado. Si en algún momento el botón pulsado es cancelar, la función "on_click" llamará a la función stop "app.stop()" que parara (cerrará) la ejecución de la interfaz y con ello la herramienta.

```
def on_click(button):
    if button == "Cancelar":
        app.stop()
    else:
        # Get values from input fields in the UI
        phi = app.getEntry("Latitud del terreno")
        shadow_percentage = app.getEntry("Porcentaje del día sin sombra")
        Z = app.getEntry("Ángulo de inclinación")
        r = app.getEntry("Orientación con respecto al sur")
        h2 = app.getEntry("Diferencia del altura del terreno")
        K = app.getEntry("Ancho del subconjunto de módulos")
        L = app.getEntry("Largo del subconjunto de módulos")
        # Some values need to be absolute value
        result = herramienta(abs(phi), shadow_percentage,
                             abs(Z), abs(r), h2, K, L)
        distancia = result[0]
        area = result[1]
        # Set result fields in the UI with the result of calculation
        app.setLabel("lresult0", distancia)
        app.setLabel("lresult1", area)
```

6 APLICACIONES

En este capítulo se procede a plantear diversas situaciones, considerando varias configuraciones de plantas fotovoltaicas. Se hará para ello uso de la herramienta y se modificarán las diferentes variables posibles. Es especialmente útil para este procedimiento la posibilidad de introducir numerosas situaciones en la entrada, y obtener en un bloque todos los resultados. Posteriormente se compararán los resultados obtenidos.

No se tomarán en cuenta las sombras producidas por otros objetos diferentes a los subconjuntos de módulos solares, aunque se pueden calcular usando las mismas fórmulas aplicadas hasta ahora.

La herramienta generada proporciona tanto la distancia entre subconjuntos como el área que ocupa cada uno. En el caso de que las dimensiones de los subconjuntos sean diferentes entre los subconjuntos, se deberán tener en cuenta al introducir los datos según indican los parámetros de la figura 4.3.

Si se tiene en consideración la potencia de los módulos y el número total de subconjuntos que se quiere instalar, se obtiene el área total ocupada por la instalación. También se puede calcular la potencia por metro cuadrado de la configuración, entre otros parámetros, y así obtener medidas tangibles para comparar los diferentes casos entre sí.
Para cada caso se compararán las magnitudes que se consideren más significativas en cada situación.

Salvo en aquellos casos en los que se indique lo contrario, en este capítulo se considerará la instalación de una planta fotovoltaica en la provincia de Sevilla, a una latitud de 37.25º, con potencia nominal 1 MWp, compuesta por módulos fotovoltaicos policristalinos de 235 Wp cada uno. El terreno se considerará plano. El rendimiento de los módulos es de 14.23% y las dimensiones de cada módulo son de 1 x 1.7 metros. Los módulos tendrán orientación sur y su inclinación será igual a la latitud en cada caso.

6.1. Variación del porcentaje del día sin sombra.

Se considerará la situación base, definida anteriormente y se plantean para este caso subconjuntos de 60 módulos, repartidos en filas de 20 paneles a lo ancho y columnas de 3 módulos. Cada subconjunto tiene una potencia de 14.1 kWp. Las dimensiones de cada uno son:

Dimensión K: 20 módulos a lo ancho, posicionados horizontalmente ocupan 20 x 1.7 m = 34 metros.

Dimensión L: 3 módulos de largo, colocados en su dimensión más pequeña ocupan 3 x 1m = 3 metros.

Se montaría tal y como se aprecia en la figura:

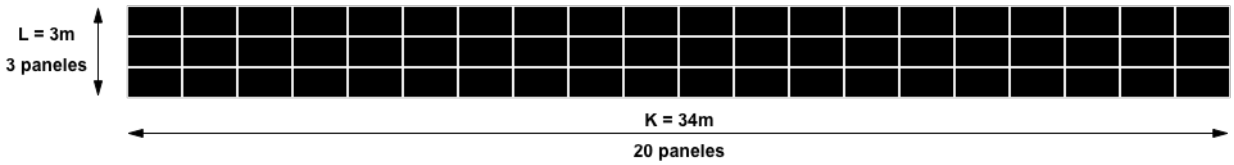

Figura 6.1. Subconjunto de dimensiones K=34 m y L = 3m

Para realizar la comparación, se procederá a variar el porcentaje del día durante el cual no habrá sombra entre subconjuntos, entre 30% y 99% y se mantienen constante el resto de las variables. Se introducen los datos en la herramienta de la siguiente manera:

Latitud del terreno:	37.25º
Porcentaje del día sin sombra:	*variable*
Ángulo de inclinación:	37.25º
Orientación:	0º
Diferencia de altura del terreno:	0 metros
Ancho del subconjunto de módulos:	34 metros
Largo del subconjunto de módulos:	3 metros

Los resultados obtenidos por la herramienta están representados en la siguiente tabla. Si se colocan 71 subconjuntos de 60 módulos para formar la planta fotovoltaica (instalando un total de 4 260 módulos), el total de área neta que se necesita calcula fácilmente a partir del área ocupada por subconjunto. Si además se tiene en cuenta la potencia pico de cada subconjunto (14.1 kWp), se puede obtener el área requerida para instalar 1kWp obtenida con la configuración, o la potencia instalada por metro cuadrado.

Al área requerida para la instalación se necesitaría añadir terreno para ubicar el resto de los elementos de la planta (inversores, transformadores, etc), pero no se tendrán en cuenta ya que no es objeto de este trabajo.

Se obtiene la siguiente tabla:

Porcentaje del día sin sombra (%)	**Distancia obtenida D (m)**	**Área de terreno ocupada por cada subconjunto (m^2)**	**Área total requerida para la instalación (m^2)**	**Área ocupada por 1 kWp de instalación (m^2)**
30	5.83	198.22	13 875.40	13.86
40	6.03	205.02	14 351.40	14.34
45	6.16	209.44	14 660.80	14.64
50	6.33	215.22	15 065.40	15.05
55	6.54	222.36	15 565.20	15.55
60	6.82	231.88	16 231.60	16.21
65	7.17	243.78	17 064.60	17.05
70	7.66	260.44	18 230.80	18.21
75	8.35	283.90	19 873.00	19.85
80	9.4	319.60	22 372.00	22.35

85	11.16	379.44	26 560.80	26.53
90	14.70	499.80	34 986.00	34.95
92	17.36	590.24	41 316.80	41.27
95	25.35	861.90	60 333.00	60.27
99	110.66	3762.44	26 3370.80	263.08

Tabla 6.1. Comparación de parámetros obtenidos frente al porcentaje del día sin sombra en Sevilla

Como es de esperar, porcentajes sin sombra cercanos al 100% dan un valor extremadamente alto (en este caso, el cálculo matemático para el 100% es de $1.6 \cdot 10^{16}$ metros. En las gráficas siguientes se muestra una representación de los valores obtenidos anteriormente de distancia y del área del terreno ocupada por subconjunto. Se han omitido los valores a partir de un valor del porcentaje de 92%

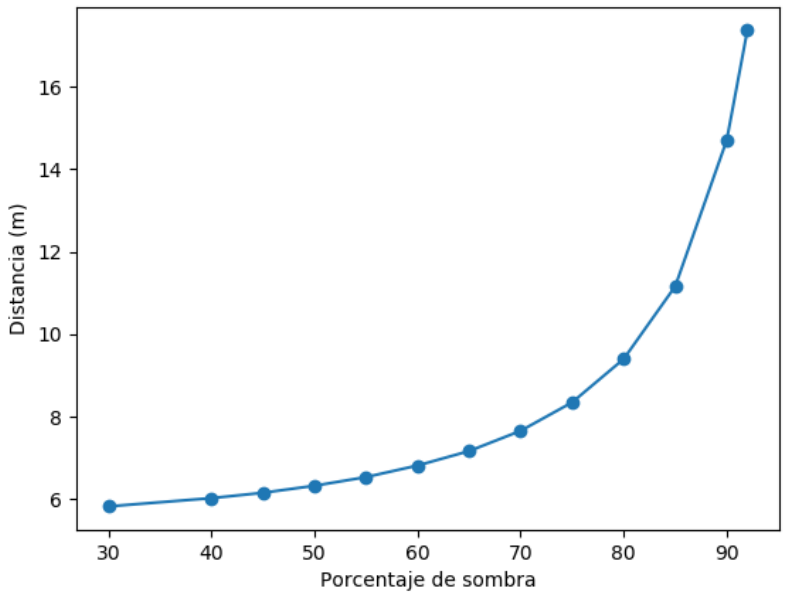

Figura 6.2. Variación de la distancia D con el porcentaje del día sin sombra.

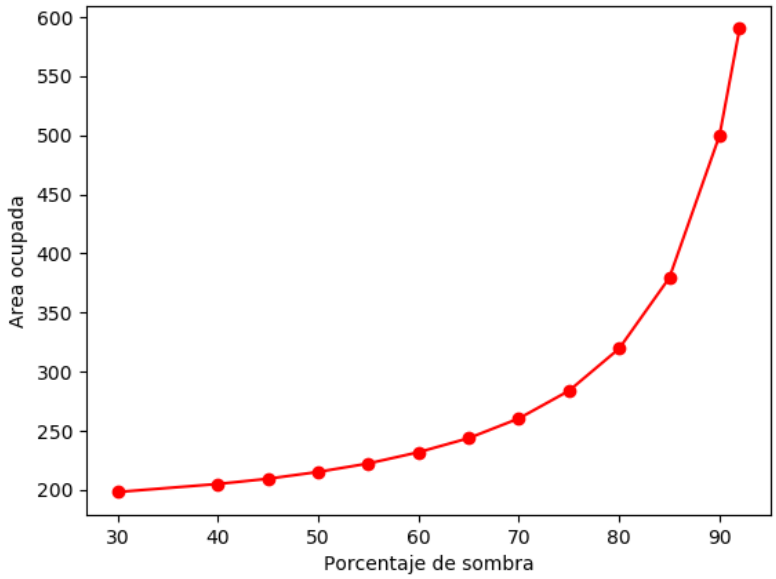

Figura 6.3. Variación del área ocupada por subconjunto con el porcentaje de sombra.

Se obtienen gráficas de tendencia exponencial, en las que a medida que los porcentajes de sombra aumenta, más distancia será necesaria para cumplir las condiciones sin sombra entre subconjuntos. No se considera práctico tomar valores muy elevados para el porcentaje de sombra, ya que la producción de energía es mayor durante las horas intermedias del día y no es necesario establecer al límite máximo las horas de luz en detrimento del aprovechamiento del terreno.

6.2. Variación del porcentaje del día sin sombra en varios ejemplos de latitudes.

Se repite a continuación el caso anterior, en el que se varía el porcentaje del día sin sombra, pero para terrenos en diferentes latitudes y comparar resultados. Las latitudes consideradas son las siguientes:

Las Canarias:	27.8º
Barcelona:	41.5º
Cercanías a Londres:	51.6º

Las características introducidas en la herramienta son las siguientes:

Latitud del terreno:	*Variable*
Porcentaje del día sin sombra:	*Variable*
Ángulo de inclinación:	*Igual a la latitud en cada caso*
Orientación:	0º
Diferencia de altura del terreno:	0 metros
Ancho del subconjunto de módulos:	34 metros
Largo del subconjunto de módulos:	3 metros

Se incluirá en la tabla siguiente los resultados de distancias entre subconjuntos para Sevilla, que provienen del caso de la sección anterior.

	Las Canarias 27.8º	**Sevilla 37.25º**	**Barcelona 41.5º**	**Londres 51.6º**
Porcentaje del día sin sombra (%)	Distancia obtenida (m)	Distancia obtenida (m)	Distancia obtenida (m)	Distancia obtenida (m)
30	4.52	5.83	6.78	11.30
40	4.62	6.03	7.03	11.88
45	4.70	6.16	7.21	12.28
50	4.79	6.33	7.44	12.78
55	4.91	6.54	7.72	13.42
60	5.06	6.82	8.08	14.24
65	5.26	7.17	8.56	15.32
70	5.53	7.66	9.21	16.77
75	5.92	8.35	10.12	18.84
80	6.50	9.40	11.51	21.97
85	7.48	11.16	13.84	27.23
90	9.45	14.70	18.54	37.80
92	10.93	17.36	22.07	42.74
95	15.39	25.35	32.66	69.60
99	62.94	110.66	145.80	324.35

Tabla 6.2. Comparación de parámetros obtenidos frente al porcentaje del día sin sombra en varias localizaciones

Se representan los resultados en la gráfica a continuación:

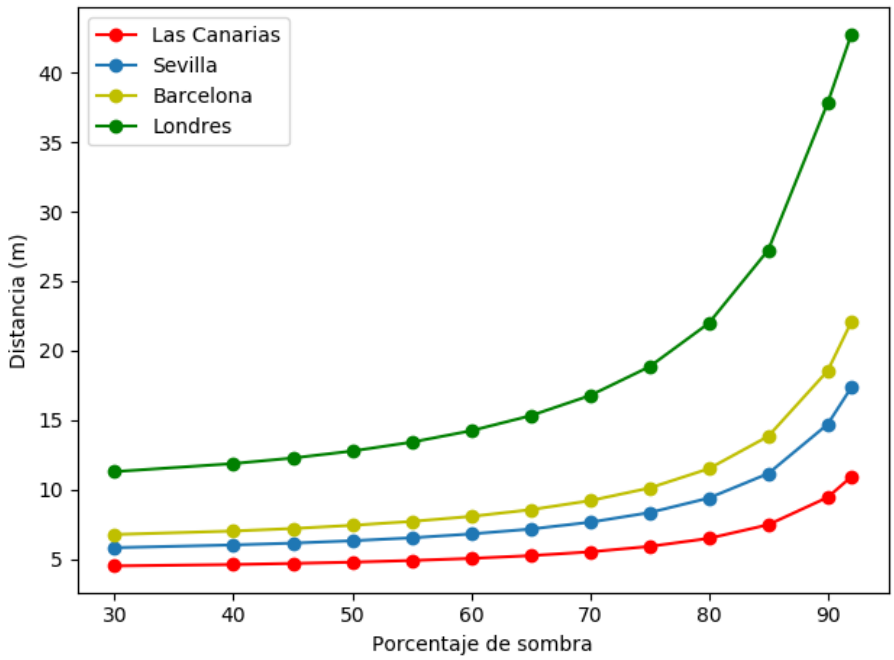

Figura 6.4. Resultados de la distancia frente al porcentaje de sombra para varias latitudes.

Como se puede observar, a medida que la latitud aumenta y los módulos se colocan en localizaciones más lejanas al ecuador, mayor es la distancia necesaria entre subconjuntos para cumplir con el porcentaje de luz solar especificado, y por tanto mayor será el área que se requerirá por kWp. En todos los casos la tendencia es exponencial, pero la diferencia en el crecimiento es considerable al variar la latitud. La influencia del porcentaje de sombra es menos para latitudes más cercanas al ecuador, mientras que en para latitudes más lejanas el aumento de distancia aumenta considerablemente una vez pasado un porcentaje superior al 70%.

6.3. Variación de la longitud de los subconjuntos, L.

En este caso se planteará una situación en la que se modifica la longitud o variable L en el código. Se considerará la planta fotovoltaica base, de 1 MWp y módulos de dimensiones 1 x 1.7 metros. Se colocarán 4260 módulos en total.

Se plantean 4 casos, con subconjuntos de 20, 40, 60 y 80 módulos, repartidos en filas de 20 paneles a lo ancho y columnas 1, 2, 3 y 4 módulos respectivamente. En todos los casos se posicionarán 20 módulos a lo ancho, horizontalmente de manera que las dimensiones de cada subconjunto son las siguientes:

Caso a)

Columnas de 1 módulos de largo, colocados en su dimensión más pequeña: 1 x 1m = 1 metro.
Esta es la dimensión denominada anteriormente como L en el código.

Filas de 20 módulos de ancho: 20 x 1.7 m = 34 metros.
Esta es la dimensión que se ha denominado anteriormente como K en el código facilitado.

Cada subconjunto tendría en total 20 paneles, con una potencia pico de 4.7 kWp y se montaría tal y como se aprecia en la figura:

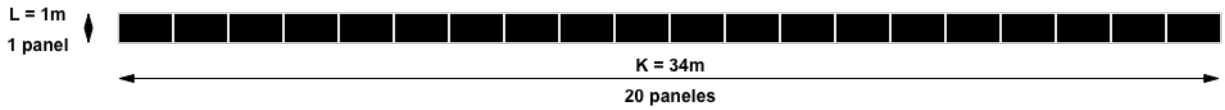

Figura 6.5. Subconjunto de 20 paneles

Para realizar la instalación se necesitará disponer de 213 subconjuntos de módulos.

Caso b)

Columnas de 2 módulos de largo, colocados en su dimensión más pequeña: 2 x 1m = 2 metros. Esta es la dimensión L en el código. Filas de 20 módulos de ancho: 20 x 1.7 m = 34 metros. Esta es la dimensión que se ha denominado anteriormente como K en el código facilitado.

Cada subconjunto tendría en total 40 paneles, con una potencia pico de 9.4 kWp y se montaría tal y como se aprecia en la figura:

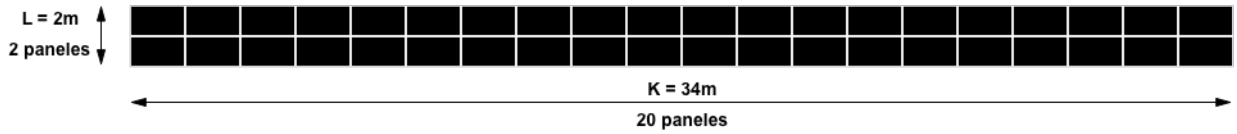

Figura 6.6. Subconjunto de 40 paneles

Para realizar la instalación se necesitará disponer de 106.5 subconjuntos de módulos.

Caso c)

Columnas de 3 módulos de largo, colocados en su dimensión más pequeña: 3 x 1m = 3 metros. Esta es la dimensión L. Filas de 20 módulos de ancho: 20 x 1.7 m = 34 metros. Esta es la dimensión K en el código facilitado.

Cada subconjunto tendrá en total 60 paneles, con una potencia pico de 14.1 kWp y se montaría tal y como se aprecia en la figura 6.1.

Para realizar la instalación se necesitará disponer de 71 subconjuntos de módulos.

Caso d)

Columnas de 4 módulos de largo, colocados en su dimensión más pequeña: 4 x 1m = 4 metros. Esta es la dimensión L. Filas de 20 módulos de ancho: 20 x 1.7 m = 34 metros. Esta es la dimensión K en el código facilitado.

Cada subconjunto tendrá en total 60 paneles, con una potencia pico de 18.8 kWp y se montaría tal y como se aprecia en la figura:

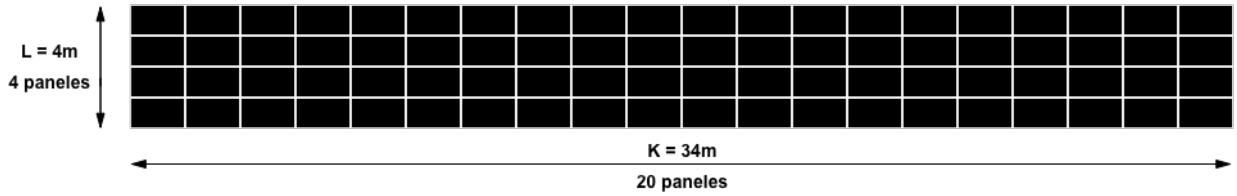

Figura 6.7. Subconjunto de 80 paneles

Para realizar la instalación se necesitará disponer de 53.25 subconjuntos de módulos.

Introduciendo en la herramienta los valores siguientes:

Latitud del terreno:	37.25º
Porcentaje del día sin sombra:	75%
Ángulo de inclinación:	37.25º
Orientación:	0º
Diferencia de altura del terreno:	0 metros

Ancho del subconjunto de módulos: 34 metros
Largo del subconjunto de módulos: *variable*

Se obtiene la siguiente tabla:

L (metros)	Distancia entre subconjuntos, D (m)	Área ocupada por un subconjunto (m^2)	Nº subconjuntos	Área total ocupada (m^2)
1	2.79	94.78	213	20188.14
2	5.58	189.57	106.5	20189.21
3	8.36	284.35	71	20188.85
4	11.15	379.14	53.25	20189.21

Tabla 6.3. Comparación de parámetros obtenidos al modificar la variable L

En todos los casos se obtienen áreas muy similares, con lo cual se deduce que la relación de distancia con altura de los paneles es prácticamente constante (como en de la fórmula práctica que se aplica generalmente, de distancia entre subconjuntos es igual a 2.5 por la altura). En este caso también es aplicable y se puede considerar eliminar este factor a tener en cuenta a la hora de tomar decisiones para el mejor aprovechamiento del terreno.

6.4. Variación del ancho de los subconjuntos, K.

Se plantea la instalación tipo usada en los casos anteriores, modificando la anchura de los subconjuntos en este caso, tomando valores desde los 8.5 metros hasta 85 para compararlos entre sí. Los valores introducidos en la herramienta son:

Latitud del terreno: 37.25º
Porcentaje del día sin sombra: 75%
Ángulo de inclinación: 37.25º
Orientación: 0º
Diferencia de altura del terreno: 0 metros
Ancho del subconjunto de módulos: *variable*
Largo del subconjunto de módulos: 3 m

Nº paneles en la horizontal	K (metros)	Nº paneles por subconjunto	Nº subconjuntos	Distancia entre subconjuntos (m)	Área ocupada por un subconjunto (m)	Área total ocupada (m^2)
5	8.50	15	284	8.35	71.00	20164
10	17.00	30	142	8.35	142.00	20164
15	25.50	45	94.6	8.35	213.00	20159
20	34.00	60	71	8.35	283.99	20163
25	42.50	75	56.8	8.35	355.99	20220
30	51.00	90	47.3	8.35	426.99	20197

35	59.50	105	40.57	8.35	496.99	20163
40	68.00	120	35.5	8.35	567.99	20164
50	85.00	150	28.4	8.35	710.00	20164

Tabla 6.4. Comparación de parámetros obtenidos al variar K

Como es de esperar, las distancias entre subconjuntos son idénticas, y la instalación ocupará aproximadamente el mismo área si el parámetro que se modifica es el ancho de cada subconjunto, ya que esto no debería afectar al área total ocupada. La definición del ancho de cada subconjunto viene determinado usualmente por las condiciones del terreno, y no genera una mayor o menos distancia de sombreado.

6.5. Variación de la inclinación de los paneles, Z.

Se tomará a continuación la situación de la planta base, pero se modificarán los valores de la inclinación de los paneles para un terreno para analizar las consecuencias en los resultados.

Se plantean para este caso subconjuntos de 60 módulos, repartidos en filas de 20 paneles a lo ancho y columnas de 3 módulos, como se indica en la figura 6.1.

Los valores introducidos en la herramienta son:

Latitud del terreno:	37.25º
Porcentaje del día sin sombra:	75%
Ángulo de inclinación:	*variable*
Orientación:	0º
Diferencia de altura del terreno:	0 metros
Ancho del subconjunto de módulos:	34 m
Largo del subconjunto de módulos:	3 m

Se tomarán ángulos de inclinación, denominada variable Z en el código, desde 0º hasta 90º, y se introducirán en la herramienta para formar la siguiente tabla:

Inclinación, Z (grados)	**Distancia entre subconjuntos, D (m)**	**Área de terreno ocupada por cada subconjunto (m^2)**	**Área total requerida para la instalación (m^2)**
0	3.00	102.00	7 242
5	3.85	130.81	9 288
10	4.67	158.63	11 263
15	5.45	185.24	13 152
20	6.19	210.44	14 941
25	6.88	234.04	16 617
30	7.53	255.86	18 166
35	8.11	275.73	19 577
40	8.63	293.50	20 839
50	9.48	322.23	22 878
60	10.03	341.16	24 222
70	10.29	349.73	24 831

Tabla 6.5. Comparación de parámetros obtenidos al variar K

Se representan los resultados en la siguiente gráfica.

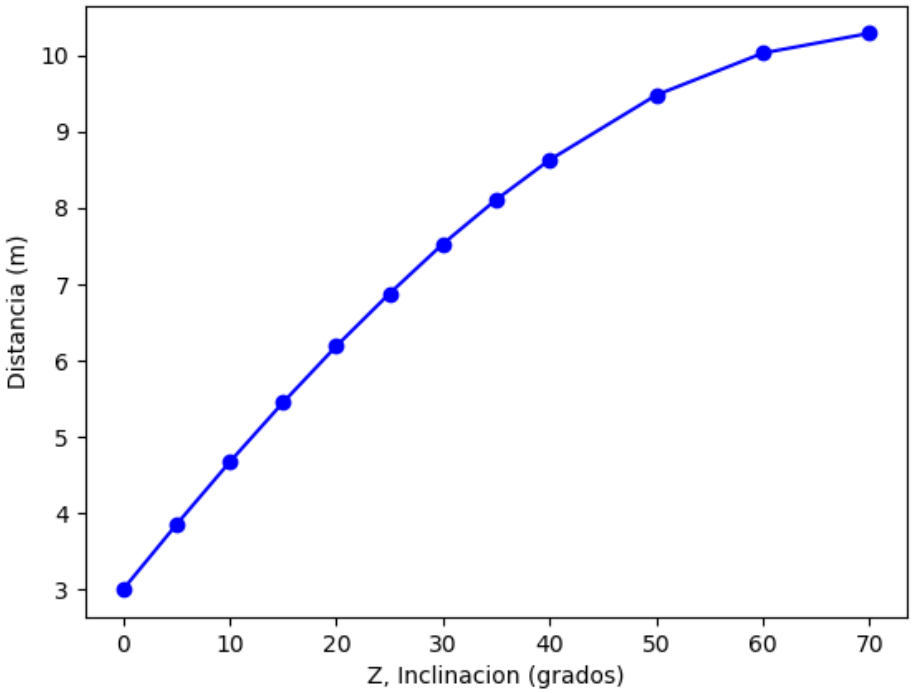

Figura 6.8. Representación de la tendencia de la distancia al aumentar la inclinación de los módulos.

Mientras menor el ángulo de inclinación de los módulos, menos área se requiere, dado que la proyección de sombra producida por los subconjuntos de módulos será menor y se requerirá menos distancia entre subconjuntos para garantizar que no se produzcan sombras entre mesas. La inclinación de los módulos está relacionada con la obtención de la mayor producción posible, acorde con la situación del terreno, y no es motivo de este proyecto analizar la orientación para obtener la mayor producción, pero desde el punto de vista del aprovechamiento del terreno y las sombras generadas, la diferencia es claramente significativa entre los diferentes casos.

6.6. Variación de la inclinación (Z) de los paneles para diferentes latitudes.

Se considera interesante aplicar el caso anterior para varias latitudes, debido a que la localización del terreno afecta en gran medida las sombras generadas, de manera que se plantea la misma situación para las siguientes localizaciones y latitudes:

Las Canarias:	27.8º
Barcelona:	41.5º
Cercanías a Londres:	51.6º

Las características introducidas en la herramienta son las siguientes:

Latitud del terreno:	*Variable*
Porcentaje del día sin sombra:	75%
Ángulo de inclinación:	*Variable*
Orientación:	0º
Diferencia de altura del terreno:	0 metros
Ancho del subconjunto de módulos:	34 metros
Largo del subconjunto de módulos:	3 metros

Se incluirá en la tabla siguiente los resultados de distancias entre subconjuntos para Sevilla, que provienen del caso de la sección anterior.

	Las Canarias 27.8º	**Sevilla 37.25º**	**Barcelona 41.5º**	**Londres 51.6º**
Ángulo de inclinación, Z (grados)	Distancia obtenida (m)	Distancia obtenida (m)	Distancia obtenida (m)	Distancia obtenida (m)
0	3.00	3.00	3.00	3.00
5	3.598	3.85	4.024	4.877
10	4.169	4.67	5.018	6.716
15	4.708	5.45	5.974	8.505
20	5.211	6.19	6.884	10.229
25	5.674	6.88	7.742	11.875
30	6.094	7.53	8.541	13.43
35	6.468	8.11	9.275	14.884
40	6.793	8.63	9.938	16.224
50	7.285	9.48	11.033	18.524
60	7.556	10.03	11.793	20.262
70	7.597	10.29	12.195	21.384

Tabla 6.6. Comparación de parámetros obtenidos frente a la inclinación de los módulos en varias localizaciones

Se representan los resultados en la gráfica a continuación:

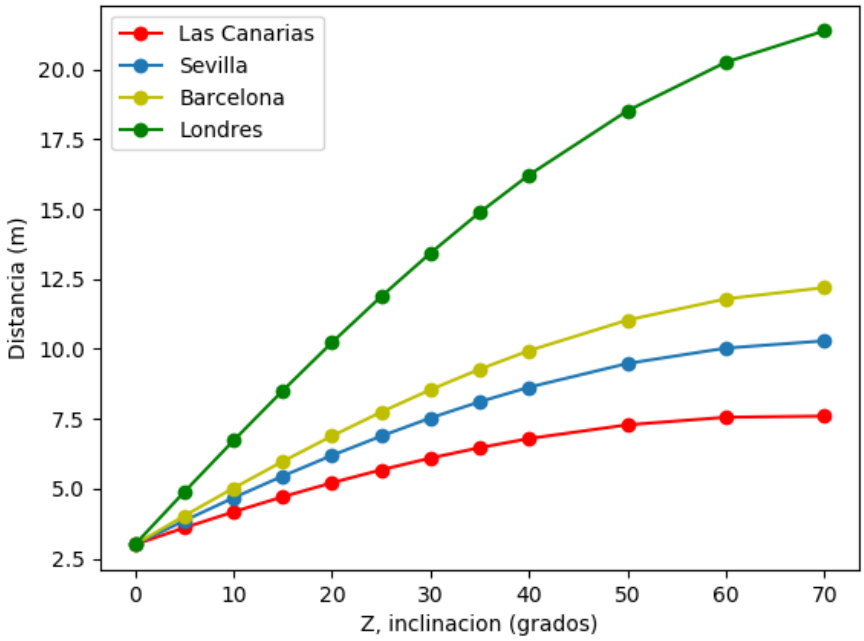

Figura 6.9. Representación de la tendencia de la distancia con la inclinación de los módulos para varias latitudes.

En la figura se aprecia claramente el crecimiento de la distancia entre subconjuntos a medida que la latitud es mayor. Especialmente llamativo el crecimiento en el caso de Londres. Esto es de esperar, debido a que la mayor inclinación de los rayos solares hace que las proyecciones de sombras sean mayores, y sea necesaria una separación mayor entre subconjuntos.

6.7. Variación de la orientación de los paneles, γ.

Se tomará a continuación la situación de la planta base, modificando los valores de la orientación de los paneles. Las orientaciones sureste y suroeste, siempre que se mantengan idénticos el resto de los parámetros, darán resultados idénticos en la distancia entre subconjuntos. Se toma como localización una planta en la provincia de Sevilla y se introducen los siguientes datos en la herramienta:

Latitud del terreno:	37.25º
Porcentaje del día sin sombra:	75%
Ángulo de inclinación:	37.25º
Orientación:	*variable*
Diferencia de altura del terreno:	0 metros
Ancho del subconjunto de módulos:	34 metros
Largo del subconjunto de módulos:	3 metros

Orientación (grados)	Distancia entre subconjuntos (m)	Área de terreno ocupada por cada subconjunto (m^2)	Área total requerida para la instalación (m^2)	Área ocupada por 1 kWp de instalación (m^2)
0	8.353	283.996	19 879.72	19.9
±5	8.916	303.146	21 220.22	21.2
±10	9.430	320.607	22 442.49	22.4
±15	9.890	336.246	23 537.22	23.5
±20	10.292	349.944	24 496.08	24.5
±25	10.635	361.596	25 311.72	25.3
±30	10.915	371.114	25 977.98	25.9
±35	11.130	378.426	26 489.82	26.5
±40	11.279	383.476	26 843.32	26.8
±45	11.360	386.225	27 035.75	27.0

Tabla 6.7. Comparación de parámetros obtenidos frente a la orientación de los módulos

A continuación, se representan los resultados en la siguiente gráfica.

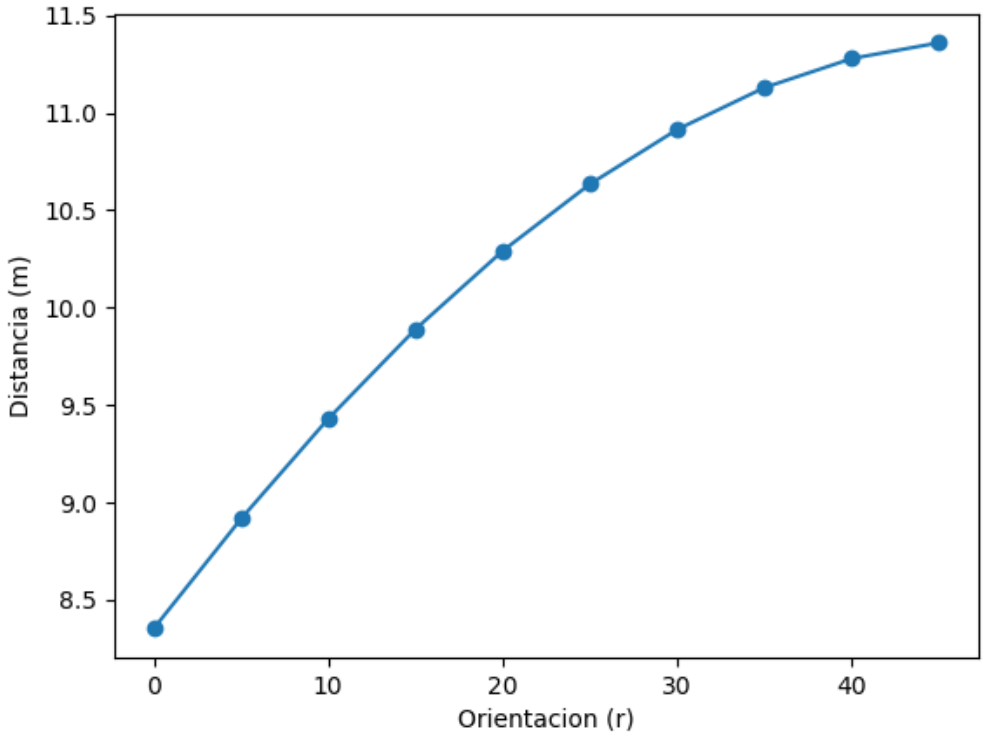

Figura 6.10. Representación de la distancia con la orientación de los módulos.

Se aprecia una tendencia exponencial de la curva, de manera que a mayor desviación de la orientación sur mayor distancia es necesaria entre subconjuntos. Las sombras tienen una componente norte-sur y otra este-oeste de manera que, al orientar la instalación hacia una dirección diferente a la sur, la componente este-oeste de la sombra aumentará y generará la necesidad de distanciar más los subconjuntos.

6.8. Variaciones de altura entre subconjuntos, h2.

Se considerará una instalación en suelo inclinado, en los que se crean terrazas para colocar la instalación base, comentada en los anteriores casos, de acuerdo con los siguientes criterios:

Latitud del terreno:	37.25º
Porcentaje del día sin sombra:	75%
Ángulo de inclinación:	37.25º
Orientación:	0º
Diferencia de altura del terreno:	*variable*
Ancho del subconjunto de módulos:	34 m
Largo del subconjunto de módulos:	*variable*

Se tomarán largos de los subconjuntos de 1, 2 y 3 paneles, para observar las diferencias entre las mismas.

Diferencia de altura entre niveles, h2 (m)	**Distancia entre subconjuntos, D (m)**		
	L = 1 m	**L = 2 m**	**L = 3m**
0	2.784	5.569	8.353
0.1	2.456	5.24	8.024
0.2	2.127	4.912	7.696
0.3	1.799	4.583	7.367
0.4	1.47	4.255	7.039
0.5	1.142	3.926	6.71
0.6	0.813	3.598	6.382
0.7	0.485	3.269	6.053
0.8	0.156	2.941	5.725
0.9	-0.172	2.612	5.396
1	-0.501	2.284	5.068

Tabla 6.8. Comparación de parámetros obtenidos frente a diferentes h2

Se representan los resultados obtenidos en la siguiente gráfica:

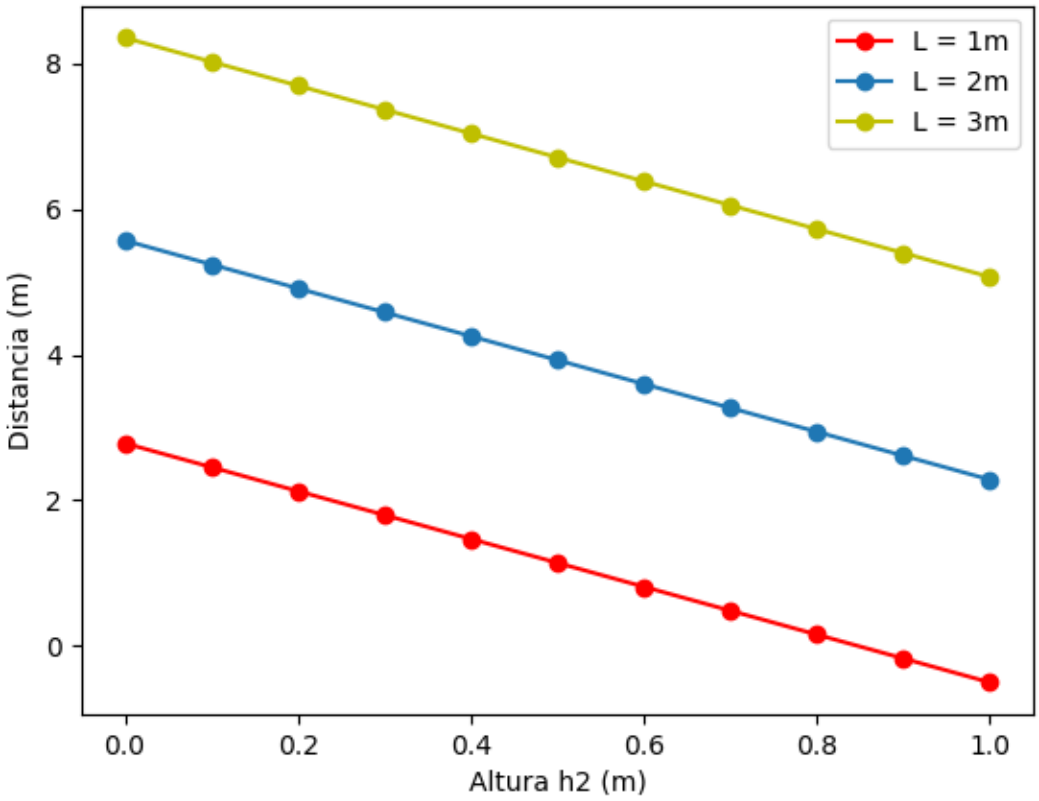

Figura 6.11. Representación de la tendencia de la distancia con la altura entre los suelos de los subconjuntos.

Lógicamente la distancia necesaria entre subconjuntos disminuye a medida que la distancia vertical entre subconjuntos aumenta. A medida que aumenta h2, disminuye h1, y esto afecta a L', la proyección horizontal de los rayos solares entre dos filas de módulos, como se representa en la figura 4.3 y en las ecuaciones que le siguen. Como resultado, la distancia D2 disminuye cuando lo hace h2, y con ellas la distancia total entre subconjuntos.

Cuando las distancias h2 se hacen muy grandes, los resultados comienzan a ser negativos como consecuencia, y esto es lo que ocurre para los subconjuntos de 1 metro en este caso, debido a que la longitud de los subconjuntos es menor.

7 CONCLUSIONES

El objetivo inicial de esta herramienta era que el usuario introdujese los parámetros para el cálculo de manera manual, a través de la interfaz gráfica. Este era el propósito principal del proyecto, y así se ha llevado a cabo, pero tras la necesidad de introducir numerosos datos para la elaboración del capítulo anterior, "Aplicaciones", se hizo evidente la mejora y ahorro de tiempo que supondría automatizar este proceso para aquellos casos en los que se desea hacer un estudio más profundo y comparación de múltiples casos. Esto permite complementar y mejorar la herramienta.

Como conclusiones obtenidas de la aplicación de la herramienta en diferentes tipos de instalaciones fotovoltaicas, cabe destacar el crecimiento exponencial de la distancia entre subconjuntos de módulos con el porcentaje del día sin sombra. Para latitudes cercanas al ecuador este crecimiento no es tan acuciado, pero a medida que nos retiramos del ecuador, para latitudes más alejadas, el crecimiento es elevado, por lo que es importante tener este factor en cuenta para no obtener valores de distancias entre conjuntos demasiado grandes. Alejar en exceso los subconjuntos por este motivo supondría contar con una instalación de menor potencia, si es que existen limitaciones en el terreno, y la obtención de energía se vería afectada, pues las horas de sol que se aprovecharían para generar energía serían las iniciales del día y las más tardías, fuera del rango de las horas solares pico, en las que hay menos valores de radiación solar y por tanto se produce menor producción de energía. La referencia [4] propone un valor del 75% en sus ejemplos resueltos, y en las aplicaciones demostradas en el capítulo 6 de este documento parece evidente que a partir de este valor las distancias comienzan a elevar su valor notablemente.

En cuanto a los casos en los que se ha modificado el ancho o alto de los subconjuntos de módulos, el área total ocupada no ha sufrido prácticamente modificaciones, y la elección de estas dimensiones vendrá definida por motivos prácticos de adaptación al terreno, o de facilidad de acceso de cara al mantenimiento de la instalación.

La inclinación de los paneles en cambio influye significativamente en las distancias entre módulos, especialmente en latitudes más alejadas del ecuador, en las que los rayos solares están más inclinados y generan mayores sombras. Tanto la inclinación como la orientación de los paneles producen crecimiento logarítmico de la distancia entre subconjuntos.

La distancia de separación en cambio presenta una tendencia lineal con la diferencia de altura entre subconjuntos. Generalmente este parámetro viene determinado por las características de la parcela, de manera que no es un parámetro que habitualmente se pueda diseñar en instalaciones en terreno, aunque sí se debe tener en cuenta en instalaciones de otro tipo. Cuando se introducen valores muy altos de las diferencias de alturas en el suelo, los valores de las distancias comienzan a ser negativos, debido a que se elimina la necesidad de separar horizontalmente los subconjuntos entre sí, ya que los conjuntos delanteros no generan sombra en ningún punto sobre los traseros.

Por otro lado, es posible trabajar sobre esta herramienta para ampliarla a diferentes configuraciones más complejas que se detallan en [4], como pueden ser estructuras con o sin seguimiento, y de uno y dos ejes.

Además, se puede completar la herramienta al introducir más parámetros de entrada en la interfaz; si se aporta el rendimiento de los módulos fotovoltaicos se puede generar la potencia de la instalación por metro cuadrado, o si se aporta el número de subconjuntos o de módulos el propio programa daría el resultado del área neta que necesita la instalación. Actualmente los datos introducidos son el largo y ancho del subconjunto, pero si se introdujesen las características de los módulos aplicados, así como la configuración en la que se montarían los subconjuntos, se simplificarían los cálculos por parte del usuario.

En todo caso, existen programas en la actualidad que realizan el cálculo de instalaciones fotovoltaicas de manera muy detallada y esta herramienta no pretende sustituir el uso de dichos programas, pero posiblemente puede aportar a la mejora de estos programas ya existentes, o sirva de complemento en aquellos casos en los que el planteamiento de una instalación se realiza de manera manual.

REFERENCIAS

[1] Quaschning V.; Hanitsch R. *Influence of Shading on Electrical Parameters of Solar Cells.* 25th IEEE Photovoltaic Specialists Conference, Washington DC, May 1996, pp. 1287-1290.

[2] Volker Quaschning and Rolf Hanitsch. *Increased energy yield of 50% at flat roof and field installations with optimized module structures*, 2nd World Conference and Exhibition on Photovoltaic Solar Energy Conversion · Vienna · Austria · 6-10 July 1998

[3] Xiu-Shui Maa, Guang-Hui Yao, Ling-Jian Yea, Xiong-Fei Zhia and Shu-Ming Zhanga.

Distance calculation between photovoltaic arrays fixed on sloping ground. Journal of Computational Methods in Sciences and Engineering 15 (2015)

[4] Wang Sicheng. *Mathematic Models and Calculation Examples for Land Usage of PV Farms.* IEC.

Documentación del lenguaje de programación Python: https://docs.python.org/2.7/